詳解

改訂版

# HTML&CSS&JavaScript 辞典

The revised edition of the detailed explanation dictionary about HyperText Markup Language, Cascading Style Sheets, and JavaScript.

大藤 幹　半場 方人　著

秀和システム

## 本書について

「詳解HTML&CSS&JavaScript辞典」が、最新の仕様とブラウザに合わせてバージョンアップしました!! コンテンツの構造を示す「HTML4.01」、表示方法を設定する「CSS2」、インタラクティブな機能を追加する「JavaScript1.5」の3つの言語から、よく使う機能をわかりやすく解説しています。基礎知識や各種チャート、仕様をまとめたリファレンスなども、ホームページ作成の強力な助っ人になるはずです。
各ブラウザの対応状況は、もはやデファクト・スタンダートとなった「Internet Explorer」を始めとして、新しいムーブメントである「Firefox」「Opera」や、根強い支持層を持つ「Mozilla」「Netscape」、また「Safari」などのMacintosh環境も含めて、最新版から普及版までフォローしています。
解説で使用したサンプルコードはサポートページで配布していますので、実際に表示させたり、コピー&ペーストで使用することもできます!

## 本書サポートページについて

**http://www.shuwasystem.co.jp/~SHOKAIdic/**

本書と合わせてご利用ください。

改訂版

# 詳解 HTML & CSS & JavaScript 辞典

The revised edition of the detailed explanation dictionary
about HyperText Markup Language,
Cascading Style Sheets,
and JavaScript.

大藤 幹　半場 方人　著

秀和システム

## 注意

(1) 本書は、著者などが独自に調査した結果を出版したものです。

(2) 本書の内容に関して、ご不明な点や誤り、記載漏れなどお気づきの点がございましたら、出版元まで書面にてご連絡ください。

(3) 本書の内容に関して運用された結果の影響については、上記(2)項 にかかわらず、著者および出版社は、いっさいの責任を負いかねますので、あらかじめご了承ください。

(4) 本書のプログラムを含むすべての内容は著作権法上の保護を受けています。著者、発行者などの書面による承諾を得ず、無断で複写、複製することは禁じられています。

## 商標

・ Microsoft、Windows、Microsoft Internet Explorerは、Microsoft Corporationの米国、およびその他の国における商標、または登録商標です。
・ Apple、Macintosh、Mac、Mac OS、Safariは、Apple Computer, Inc.の米国およびその他の国における商標、または登録商標です。
・ Mozilla、Firefoxと、それぞれのロゴは、Mozilla Foundationの商標です。
・ Netscape、Netscape Navigator、Netscape Communicatorは、米国Netscape Communications社の商標です。
・ Operaは、Opera Software ASAのノルウェーおよびその他の国における商標、または登録商標です。
・ JavaTMおよびすべてのJava関連の名称は、Sun Microsystems,inc.の米国およびその他の国での商標、または登録商標です。
・ その他、ブランド名、製品名、および会社名は、一般に各メーカーの商標、または登録商標です。
・ 本書の中では通称、またはその他の名称で表記していることがありますので、ご了承ください。

# はじめに

CSS1の仕様が勧告として公開されたのは1996年、HTML4.0が公開されたのは翌1997年のことです。その後、CSS2が公開され、HTMLはHTML4.01、XHTML1.0、そしてXHTML1.1へと進化しました。一方、ブラウザも次々と改良が行われ、比較的新しいバージョンでは、スタイルシートもほぼ問題なく利用できるようになっています。現在では、標準的なHTMLとCSSの仕様に基づいて制作すれば、異なるブラウザでも同じ表示結果を得られるような状況に、ほぼなっているのです。

本書は、将来的にはXHTMLに移行することも見据えた上で、標準的な仕様に沿ったホームページを制作できるような内容となっています。また、より多くの人が利用できるようなホームページにするための、Webアクセシビリティに関するヒントも多く掲載しました。本書が、よりよいホームページ制作のための一助となれば幸いです。

大藤　幹

JavaScriptに対応した初めてのブラウザ、Netscape Navigator2.0が発表されてから、かれこれ10年近くたたちます。当時、サイトを構築する上でJavaScriptを使うことは、なにか特別な感じがありましたが、今では、ごく普通のこととなっています。また、JavaScript自身も、ここ数年、以前ほど大きな仕様の変更はなく、安定しています。

このような状況で、今後JavaScriptに必要なことは、CSS2やDOMを始めとした、仕様が標準化されている技術との連携だと考えています。現在では、Internet ExplorerやNetscape Navigatorだけでなく、Firefox、Safariといった、ほとんどのユーザーが使用しているブラウザが、上記のような多くの標準化された技術に対応しています。そしてJavaScriptは、標準化された仕様に準拠した記述を行うことによって、これらの技術と連携をとることが可能です。

今回本書をまとめるにあたり、今まで以上に、標準化された技術との連携に重点を置いて、改定作業を行うように心がけました。そしてそれは、JavaScriptを使ってウェブサイトを構築する上で、大きな利点になるはずだと信じています。

半場　方人

# 本書の使い方

**書式**
タグやスクリプトの書式。青い斜体文字は、数値などの値を設定する部分です。HTMLパート・CSSパートでは、下の表で値の解説をする場合があります。

表示例の解説。ブラウザ名か、サンプルの動きを表しています。

サンプルソースのブラウザ表示例。効果が一目瞭然です。

**解説**
使用方法や注意点などを、文章でわかりやすく説明します。

**サンプル**
改変可能なソースを掲載しています。赤文字·青文字は「書式」に対応している部分です。サンプルはすべてサポートページにて配布いたします(ただし、紙面の都合で折り返しや改行などが異なっていることがあります)。【〜】はサンプルの一部を抜き出している場合の部分名か、ファイルが複数ある場合のファイル名です。

**タイトル**
できること·したいことから、項目を探すことができます。

CSS

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

## 行間を設定する

**line-height:** *行の高さ*

| 行の高さ | normal・数値・単位付きの数値・% |
|---|---|

Internet Explorer 6.0 / Firefox 1.0

行間の設定

line-height: normal

行の高さは文章の読みやすさに影響します。狭すぎても広すぎても読みにくくなります。行数が少ない場合は行間が少なくても比較的読みやすく、1行の文字数が多い場合は行間が狭いとかなり読みにくくなります。

line-height: 1.5

行の高さは文章の読みやすさに影響します。狭すぎても広すぎても読みにくくなります。行数が少ない場合は行間が少なくても比較的読みやすく、1行の文字数が多い場合は行間が狭いとかなり読みにくくなります。

line-height: 180%

行の高さは文章の読みやすさに影響します。狭すぎても広すぎても読みにくくなります。行数が少ない場合は行間が少なくても比較的読みやすく、1行の文字数が多い場合は行間が狭いとかなり読みにくくなります。

テキストを設定する

line-heightプロパティは、行の高さを設定します。
単位を付けないで数値だけを指定した場合は、フォントサイズにその値を掛けた高さに設定されます。%による指定も、フォントサイズに対する割合になります。マイナスの値は指定できないことになっていますので、注意してください。

**Sample**

【CSS】
```
#sample1 { line-height: normal }
#sample2 { line-height: 1.5 }
#sample3 { line-height: 180% }
em {
  color: #ff3300;
  background-color: #ffffff;
  font-style: normal
}
```

212 テキスト

本書は、主に「Windows XP」「Internet Explorer 6.0」「Firefox 1.0」の日本語版で、動作確認とイメージの作成を行っております。例外の場合は、それぞれに表記してあります。フォントなどの表示設定は、インストール直後の状態になっていますが、紙面の都合により「ステータス バー」などの一部のツールを非表示にしてあります。

**対応ブラウザ**
サンプルをブラウザで動作確認した結果を表示しています。色が薄くなっているものは「一部未対応」、空欄のものは「未対応」です。詳細は「解説」をお読みください。

【HTML】

```
<h1>行間の設定</h1>

<h2>line-height: <em>normal</em></h2>
<p id="sample1">
行の高さは文章の読みやすさに影響します。狭すぎても広すぎても読みにくくなります。行数が少ない場合は行間が少なくても比較的読みやすく、1行の文字数が多い場合は行間が狭いとかなり読みにくくなります。
</p>
<h2>line-height: <em>1.5</em></h2>
<p id="sample2">
行の高さは文章の読みやすさに影響します。狭すぎても広すぎても読みにくくなります。行数が少ない場合は行間が少なくても比較的読みやすく、1行の文字数が多い場合は行間が狭いとかなり読みにくくなります。
</p>
<h2>line-height: <em>180%</em></h2>
<p id="sample3">
行の高さは文章の読みやすさに影響します。狭すぎても広すぎても読みにくくなります。行数が少ない場合は行間が少なくても比較的読みやすく、1行の文字数が多い場合は行間が狭いとかなり読みにくくなります。
</p>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

紙面の都合で折り返していることを表しています。

font：「フォント」の「フォント関係をまとめて指定する」(P.210)

**リンク**
文法的に関連がある部分や、応用する場合に必要となる箇所を参照することができます。

テキストを設定する

**ツメ**
分類された機能を一言で表しています。機能ごとに探す時に便利です。

**コラム**

**line-heightプロパティのバグに注意！**

Internet Explorer 3.XやNetscape Navigator 4.Xには、line-heightプロパティに関するバグがたくさんあります。たとえば、Internet Explorer 3.Xの場合は単位を付けない数値を指定するとピクセル数として認識されますし、Netscape Navigator 4.Xでは絶対的な単位を指定していると印刷で問題が発生する場合もあります。

中でも特に注意する必要があるのが、Netscape Navigator 4.Xではline-heightを指定した範囲に画像が含まれていると、テキストの上に画像が重なって表示されるというバグです。Netscape Navigator 4.Xのユーザーから「文字の上に画像が重なって読めない」という苦情が来たら、まずはline-heightを疑ってみましょう。

ページの項目やその前後の項目に関連した注意点やワンランク上のテクニックなどを紹介しています。「コラム」の他に「TIPS」「重要」「注意」があります。

テキスト 213

**柱**
本書の項目は、HTMLパート・CSSパートは機能ごと、JavaScriptはオブジェクト名ごとに分類されています。その項目名で探す時の目安になります。

# 目次

## HTMLパート

### HTMLについて



### 基本的な内容



### 文書情報



### テキストの種類

## スタイルとレイアウト



## リンク



## リスト



## テーブル

## 画像とマルチメディア



## フォーム



## フレーム

## スクリプト



## リファレンス



# CSSパート

## CSSについて



## CSSの組み込み



## CSSを適用する対象



## フォント

## frame オブジェクト



## document オブジェクト



## history オブジェクト



## location オブジェクト



## Link オブジェクト・Anchor オブジェクト

## Formオブジェクト



## Areaオブジェクト



## Imageオブジェクト



## Layerオブジェクト



## スタイルシート

## ビルトインオブジェクト

### Dateオブジェクト



### Mathオブジェクト

## stringオブジェクト



## Arrayオブジェクト



## functionオブジェクト



## Objectオブジェクト

## Booleanオブジェクト



## Numberオブジェクト



# その他

## 複数のオブジェクトで使用できるプロパティ・メソッド



## ビルトイン関数(top-level関数)



# リファレンス

## JavaScript付録コラム



## 索引パート



## 巻末付録

詳解
HTML&CSS&
JavaScript
辞典 改訂版

The revised edition of the detailed explanation dictionary
about HyperText Markup Language,
Cascading Style Sheets,
and JavaScript.

# HTML

# HTML4.01とは？

インターネットのブームは、Mosaicという画像を表示できるWebブラウザと共に突然やってきました。

そして、それに続いてNetscape NavigatorというWebブラウザが登場しました。Netscape Navigatorは、プラグイン機能でさまざまなデータの取り扱いを可能にし、フレーム機能やJavaScriptなどのスクリプト言語も利用できるようにしていきました。

そのうち、途中から登場してきたInternet ExplorerとNetscape Navigatorは、激しいシェア争いのために、次々と独自の機能を追加するようになっていきます。そして、ユーザーは「新しい機能」をこぞって使うという時期が、しばらく続いていました。当時は単純にアクセス数が多ければ多いほどよいと思われていた時代で、新しい機能を使って「こんなこともできる!」というページを作るだけで、アクセス数が多くなったことを記憶しています。

しかし、しばらくすると、それによる弊害も現れてきました。常に最新のWebブラウザを利用していないと、見ることのできないページが多くなってしまったのです。当時のWebサイトには、トップページに「○○○のバージョンX.XX以上でご覧ください」などと書かれていたことが、それを象徴しています。つまり、本来は多くの人に見て欲しいはずのWebページが、ユーザーを限定してしまう方向へと進んでしまったわけです。

ちょうどその頃、WWWの標準化を行っているW3C(World Wide Web Consortium)が、HTML4.0という仕様を発表しました。この仕様のもっとも重要な点は、「今後はHTMLから表示方法やレイアウトなどの『表現』に関する部分を排除していく」という明確な方針を示したことです。同時に、HTMLにスタイルシートを組み込むための仕組みも正式に導入され、Webコンテンツの表示方法については、スタイルシートを使用して制御することが推奨されました。

では、HTMLから「表現方法」に関する部分を取り除くと、どうなるのでしょうか?たとえば、「見出し」の部分を、見出しを意味するh1要素(<h1>～</h1>)としてタグ付けした場合と、font要素(<font>～</font>)で大きさや色だけを指定した場合を考えてみましょう。

もし、それがh1要素であれば、ブラウザはその環境に合わせた表現をすることができます。たとえば、文字の大きさが変えられないブラウザであれば、見出しをセンタリングしたり改行するなどして表現できますし、音声ブラウザであれば読み上げる速度や声の種類を変えることによって表現することもできます。

しかし、これがfont要素であれば、フォントサイズや色が表現できない環境では、その意味がまったくなくなってしまうのです。

つまり、「HTMLではそれが何であるかだけを示し、表示方法を指定(限定)しない」ということは、「さまざまな環境でそれに合わせた表現方法を使って情報を伝えること

ができる」ということになります。これによって、HTMLで書かれた情報が、白黒の画面でも、テキスト端末でも、音声でも、点字でも、その環境に応じて利用することができるようになるというわけです。そして、一般的なパソコンなどの環境では、スタイルシートを利用することで、今まで以上の豊かな表現も可能になるのです。

しかし、急に「HTMLから表示に関する部分を取り除きましょう」と言われても、ブラウザがスタイルシートに対応していなければ、すぐにそうするわけにもいきません。そこで、HTML4.0では、今まで利用されてきた「表示を制御する要素」も「非推奨」としてではありますが、使うことができるようになっています。

HTML4.0には、「非推奨」の要素を使うかどうか、またフレームを利用するかどうかによって、3パターン(3種類のDTD)が用意されているのです。したがって、制作者は、そのサイトの目的やターゲットに応じて、過去の「表示を制御する要素」も引き続き使うこともできますし、逆にそれらを排除して、より多くの環境で利用できるようなWebページを制作することもできます。

HTML4.01は、HTML4.0の仕様書にあった不具合を修正し、既存のブラウザとの互換性を考慮して多少の変更を行ったもので、本書はこのHTML4.01の仕様を元に解説しています。実際には、HTML4.01はXMLで再構築されてXHTML1.0となり、さらにそれがモジュール化されてXHTML1.1へと進化していますが、基本的な要素や属性の使い方はHTML4.01から変わっていません。「非推奨」の要素や属性を使わずに、正しい書き方さえしておけば、XHTMLへの移行も意外に簡単です。

## 要素と属性

ここでは、HTMLを理解する上で欠かせない用語である「要素」や「属性」などについて簡単に説明します。

まず、次の文章を見てください。

**HTMLの用語について**

**HTMLで使われる用語を正しく理解していると、**
**文法やその構造がいっそう理解しやすくなります。**

上の例では、まず最初に「HTMLの用語について」という見出しがあって、それについて書かれた段落が続いています。

これは、人間が見ればわかることなのですが、コンピュータにとってはそう簡単に識別できるものではありません。そこで、それぞれの構成要素を、その役割を表す目印を付けて示すことにします。上の例では、「見出し」と「段落」を示せばよいので、その範囲も明確にするために、その開始部分と終了部分に目印を付け、終了の目印にはそれがわかるように「/」も付けてみましょう。実際に目印を付けてみたものが次の例です。

<見出し>HTMLの用語について</見出し>

<段落>HTMLで使われる用語を正しく理解していると、
文法やその構造がいっそう理解しやすくなります。</段落>

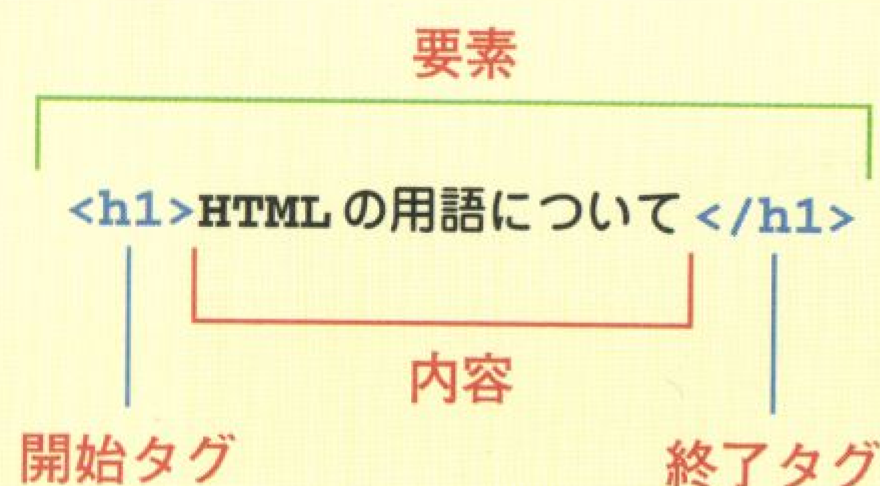
このように、あらかじめ構成要素を表す目印を決めておけば、コンピュータもその部分が何であるかを簡単に判断することができるようになります。そして、たとえば見出しであれば大きな文字で表示するなど、状況に応じた表現でそれを示すことができるようになります。

上の例では、目印を日本語で付けましたが、それを英語にしてみると「見出し」は「heading」、「段落」は「paragraph」になります。見出しには大見出しや小見出しなどの種類がありますので、それを数字で表すことにして、それぞれの目印を単語の頭文字で示してみると、次のようになります。

```
<h1>HTMLの用語について</h1>

<p>HTMLで使われる用語を正しく理解していると、
文法やその構造がいっそう理解しやすくなります。</p>
```

完全ではありませんが、見事にHTMLになりました。HTMLでタグを付けるということは、このようなことなのです。上の例でいう「目印」が、HTMLの「タグ(tag)」というわけです。そして、タグによって役割と範囲を示された構成要素そのものが、HTMLの「要素(element)」なのです。HTMLで要素といえばタグも含めた全体を指しますが、タグを除いた部分を示す場合には、それを「内容(content)」と表現します。




次に、要素の表し方について、もう少し詳しく説明します。HTMLでは、一般に要素は開始タグと終了タグで囲って示しますが、開始タグは「<要素名>」のように、終了タグは「</要素名>」のように前に「/」を付けて記述します。この場合、HTML4.01では要素名を大文字で書いても小文字で書いてもかまいません。ただし、XHTMLでは要素名をすべて小文字で書くことになっていますので、注意してください(本書では、小文字に統一しています)。

また、要素には内容を持たないものもあります。たとえば、画像を表示するためのimg要素や、改行のためのbr要素などがそれにあたります。これらは、「空要素」と呼ばれ、開始タグのみでそれを表します。終了タグは必要ありませんので、書かないようにしてください。

次に「属性（attribute）」について説明します。属性は、要素の開始タグの中に指定して、その要素の性質や特性を示すために使用されます。基本的には「属性名="値"」の形式で表し、スペースで区切って複数指定することもできます。その場合の指定順序は問いません。HTML4.01の場合は、要素の場合と同様に属性名は大文字でも小文字でもかまいませんが、XHTMLの場合は小文字で書くことになっています。また、HTML4.01では属性の値を囲う引用符（"）を、条件によっては省略できることになっています。しかし、これもXHTMLでは省略できませんので、常に省略しないで書いておいたほうがいいでしょう（HTML4.0でも省略しないことが推奨されています）。なお、引用符には「"」だけでなく、「'」を使うこともできます。

## ブロックレベル要素とインライン要素

文書中で使用される要素の多くは、ブロックレベル要素とインライン要素に分類することができます。また、インライン要素の中には置換要素と呼ばれるものがあります。

### ブロックレベル要素

その名の通りブロック（ひとつのまとまった単位）として表される要素で、見出し・段落・リスト・テーブルなどが該当します。一般的に、その前後は改行されます。

| | | | | |
|---|---|---|---|---|
| address | blockquote | center | div | dl |
| fieldset | form | h1～h6 | hr | noframes |
| ol | p | pre | table | ul |

### インライン要素

文章の一部分として含まれる要素で、リンク・強調・略語などが該当します。

| | | | | |
|---|---|---|---|---|
| a | abbr | acronym | applet | b |
| basefont | big | br | button | cite |
| code | dfn | em | font | i |
| iframe | img | input | kbd | label |
| object | q | s | samp | select |
| small | span | strike | strong | sub |
| sup | textarea | tt | u | var |

### 置換要素

インライン要素のうち、表示される時に要素自体があるもので置き換わるような要素を指します。

| | | | | |
|---|---|---|---|---|
| img | input | object | select | textarea |

※ins 要素と del 要素は、指定する範囲に応じてブロックレベル要素とインライン要素のいずれにもなります。

## 色の指定方法

HTML で色を指定するには、2 通りの方法があります。実際の色とその値については、巻末付録「カラーチャート 1 〜 3」を利用すると便利です。

### 16 進数で指定する

「#」記号に続けて、RGB（Red, Green, Blue）の各値を 2 桁ずつの 16 進数で示します。たとえば、赤を指定する場合には「#ff0000」となります。

### 色の名前で指定する

HTML4.01 では、基本的な 16 色（巻末付録「カラーチャート 1 ： HTML4.01 で名前が定義されている色」を参照）については、色を名前で指定することができます。たとえば、赤の場合は「red」と指定できます。この場合、大文字と小文字は区別されません。

※色を指定する場合は、HTML を使用せずに、スタイルシートを利用することが推奨されています。

## ファイルの位置指定 - 絶対 URL と相対 URL -

たとえば、ある部分をリンクさせる場合はリンク先の HTML ファイルの位置を、画像を表示させたい場合にはその画像の位置を指定する必要があります。HTML では、この位置を URL で示しますが、それには 2 通りの方法があります。

### 絶対 URL

これは Web ブラウザでページを見ている時にアドレスバーなどに表示されている、「http://」で始まる形式の指定方法です。この方法で指定すると、そのデータの転送方式やサーバー、サーバー内での位置まで完全に指定することになります。

一般に、自分のサイト内から他のサイトへとリンクする場合など、他のサイトのファイルに対して使用される形式です。

【例】 `http://www.w3.org/`

## 相対URL

相対URLは、同じサイト内で参照を行う場合など、同じディスク上のファイルを参照する場合に利用される形式です。この場合、現在のファイルの位置を基準として、ディレクトリ（フォルダ）の階層の上下を示すことによって位置を示します。

自分でホームページを作成している場合など、オフラインの状態でも利用できるようにするためには、この方法で指定してください。

相対URLの指定方法は、自分より下の階層にあるファイルの場合は、そのディレクトリ名からファイル名までを順に「/」で区切って記述していきます。上の階層を示すには、ひとつ上を示すごとに「../」を付けて指定します。

・同じ階層（ディレクトリ）のファイルを示す場合：

```
ファイル名
```

・ひとつ下の階層（同じ階層にあるディレクトリ内）のファイルを示す場合：

```
ディレクトリ名/ファイル名
```

・ふたつ下の階層のファイルを示す場合：

```
ディレクトリ名/ディレクトリ名/ファイル名
```

・ひとつ上の階層のファイルを示す場合：

```
../ファイル名
```

・ふたつ上の階層のファイルを示す場合：

```
../../ファイル名
```

・ひとつ上の階層の別ディレクトリのファイルを示す場合：

```
../ディレクトリ名/ファイル名
```

# iモード対応のホームページ作成

「iモードサービス」を利用すると、携帯電話からインターネットに接続して、Webページを見ることができます。しかし、その場合に見ることのできる内容には、いくつかの制限があります。

実際にiモードから見てもらうためには、iモード端末専用のページを作成する必要があると考えたほうがよいでしょう。

具体的な制限事項としては、次のようなものがあります。

・使用できる要素や属性が限られている
（P.173のリファレンス「iモード端末で利用できる要素・属性一覧」を参照）
・表示可能な文字コードはシフトJISのみ
・表示可能な画像はGIF形式（JPEGに対応した機種もある）
・スクリプト言語には未対応

また、全機種で利用できるようにするためには、さらに以下の基準を守る必要があります。

・画像は2階調のGIF形式を使用する
・画像のサイズは最大で94×72ピクセル以内にする
・1ページのサイズは5KB以内にする（2KB以内を推奨）
・表示画面は全角で横8文字×縦6行

その反面、iモード端末では、次のようなことが可能になっています。

・独自の「絵文字」が利用できる
・半角カナが問題なく利用できる
・リンクを利用して電話をかけることができる
（URLとして「tel:～」形式が利用可能」）

【例】`<a href="tel:03-1234-5678">～</a>`

・ショートカットキーとして「0～9, #, *」が単独で利用できる
（ただし、端末によっては「#, *」は使用できない）

【例】`<a href="http://～" accesskey="1">～</a>`

# HTMLのバージョンを示す

## <!DOCTYPE ~ >



そのHTML文書が、HTMLのどのバージョンの決まりに従って書かれているのかを示します。

HTMLには、HTML3.2・HTML4.01・XHTML1.0・XHTML1.1などの異なるバージョンがあり、HTML4.01とXHTML1.0は、さらに「Strict」「Transitional」「Frameset」という3つの種類に分かれています。使用するバージョンや種類によってお決まりの書き方がありますので、そのまま文書の先頭に配置してください。ここで使用するバージョンを示したら、そのバージョンの決まりに従ったHTMLを書く必要があります。本書の内容に沿ったページを作成するのであれば、以下のサンプルで示すようにHTML4.01の「Transitional」を指定しておけば基本的に問題ないでしょう。

なお、<!DOCTYPE>は要素を表すタグではなく「文書型宣言」というもので、正式なHTMLには必須のものです。

### Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
  ∫
</head>
<body>
  ∫
</body>
</html>
```

### TIPS

**新しいブラウザは<!DOCTYPE>の書き方で表示が変わる！**

Internet ExplorerのWindows版の6.0やMacintosh版の5.X以降を始めとする比較的新しいブラウザ(Firefox、Mozilla、Netscape 6.0以降、Opera、Safariなど)には、ふたつの表示モードがあります。ひとつは標準的な仕様に従った表示をする「標準モード」で、もうひとつはこれまでの各社ブラウザの独特な表示方法との互換性を持たせた「互換モード」です。このふたつのモードは、<!DOCTYPE>がどのように書かれているかで自動的に切り替わりますので、注意してください。各ブラウザで切り替わる条件が多少異なってはいますが、HTML4.01 StrictとHTML4.01 TransitionalでURL部分を省略しない場合には「標準的モード」に、HTML4.01 TransitionalでURL部分を省略した場合と<!DOCTYPE>自体が書かなかった場合には「互換モード」になるという点では共通しています。本書では、それぞれのモードで表示結果が異なる部分については、その都度解説していますので、参考にしてください。

TIPS

## HTML4.01の種類と<!DOCTYPE～>の書き方

以下に、本書の内容に沿ったページを作成する場合に使用されると思われるHTML4.01の種類と、それを使用した場合の<!DOCTYPE>の書き方を示します。紙面の都合でそれぞれ2行になっていますが、1行にしても問題ありません。また、各2行目のURLの部分("http://～.dtd")は省略することもできます。

**・HTML4.01 Strict**

本来の理想的なHTML4.01を書く場合には、この<!DOCTYPE>を指定してください。ただし、HTML4.01で非推奨とされているレイアウトなどの見栄えを表現する要素や属性とフレームは使うことができません。また、各要素を配置できる位置や順序などについても、厳しい制限があります。これに沿ったページを作るには、本書で紹介している以上の知識も必要になります。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
```

**・HTML4.01 Transitional**

HTML4.01で非推奨とされている要素や属性も使用可能で、各要素を配置できる位置や順序についても自由度の高いHTMLです。ただし、インラインフレーム以外のフレームは使用できません。一般的に考えるともっとも利用しやすい種類なのですが、上記のHTML4.01 Strictを採用することが難しい場合に利用するための、暫定的なバージョンであるということもできます。本書では、基本的にHTML4.01 Strictに沿った解説をしていますが、細かい部分でそれに従っていない箇所が生じる場合もありますので、この<!DOCTYPE>を使用することをお勧めします。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
```

**・HTML4.01 Frameset**

フレームを使用する場合に指定する<!DOCTYPE>です。インラインフレーム以外のフレームも使用できるという点を除けば、HTML4.01 Transitionalと基本的には同じものです。

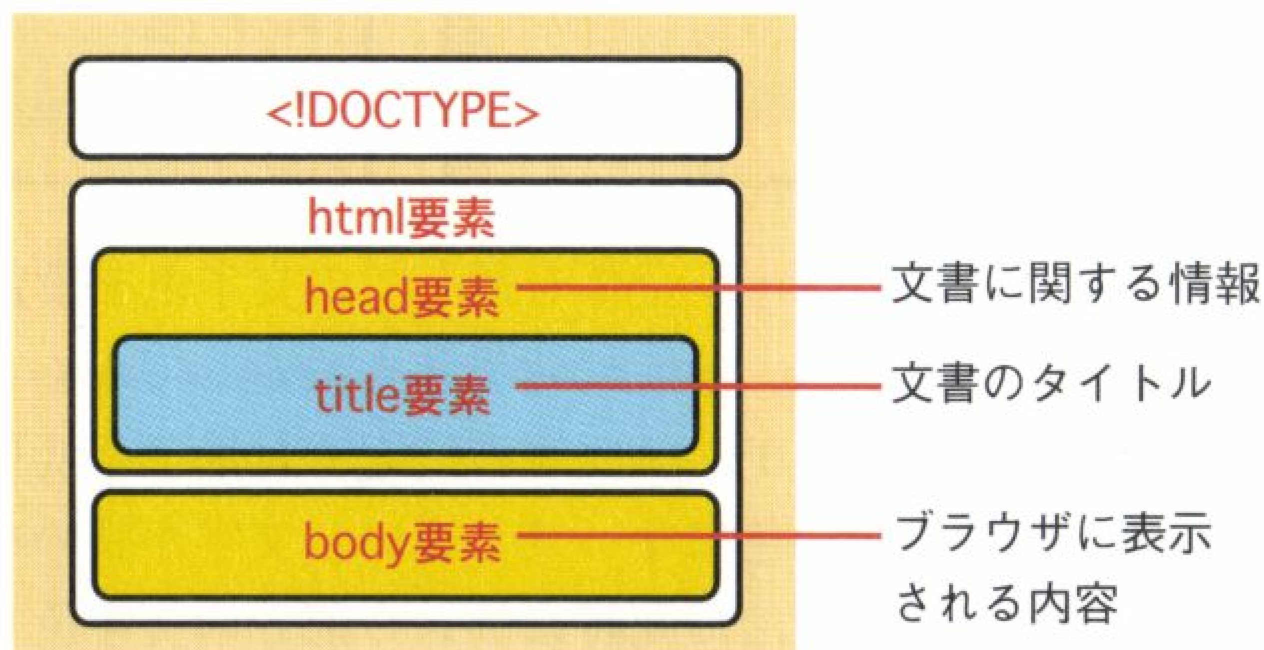
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
 "http://www.w3.org/TR/html4/frameset.dtd">
```

# 最低限必要な要素

**<html>～</html>**
**<head>～</head>**
**<title>～</title>**
**<body>～</body>**



**HTML文書の構造**




HTML文書の先頭にはHTMLのバージョンを示す<!DOCTYPE～>を配置しますが、その後には、最低限ここで説明する4つの要素が必要となります。
まず、<!DOCTYPE>以降の全体をhtml要素(<html>～</html>)で囲うようにして、その中にhead要素(<head>～</head>)とbody要素(<body>～</body>)を順にひとつずつ配置します。
head要素の中にはその文書に関する情報を入れ、body要素の中には実際にブラウザで表示される内容を書いていくことになります。また、head要素の中には、その文書のタイトルを示すtitle要素を必ずひとつだけ入れておく必要があります。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
      …この文書に関する情報…
<title>この文書のタイトル</title>
      …この文書に関する情報…
</head>
<body>
      …実際に表示される内容
</body>
</html>
```

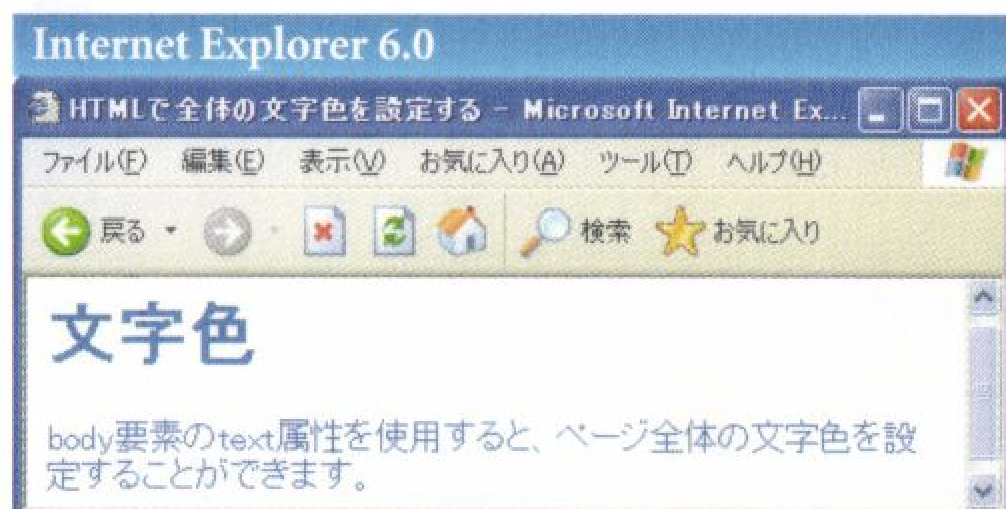

コラム「何語で書かれているかを示す」(P.26)

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac
i-mode

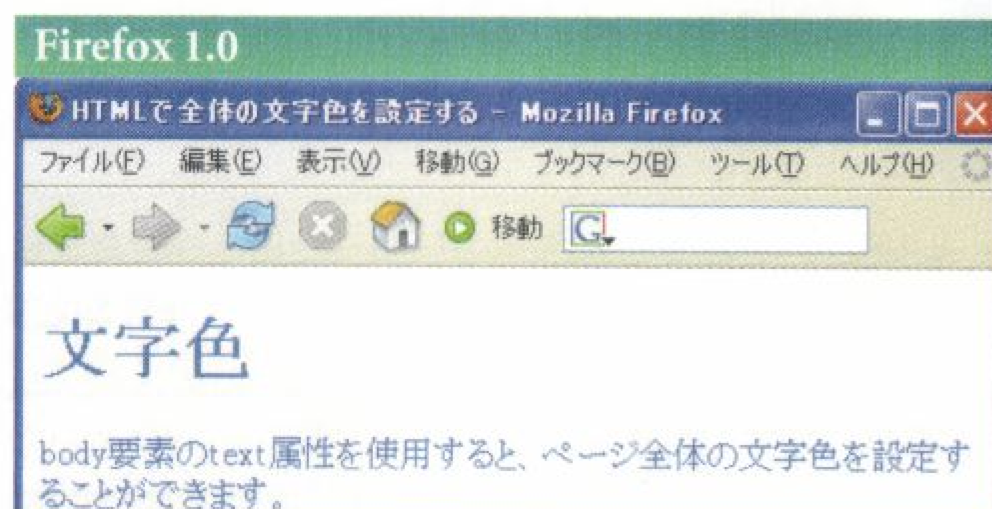
# 全体の文字色を設定する

## <body text="色指定">～</body>

Internet Explorer 6.0



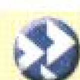

Firefox 1.0




body 要素の text 属性は、ページ全体を通しての文字色を設定します。
文字色を設定する場合には、同時に背景色も設定しておくようにしてください。そうしておかないと、ユーザーがブラウザの設定で背景色を変更している場合などに、文字色と背景色が同じ系統の色になってしまい、文字が読めなくなってしまう可能性があります。また、同様の理由で同時にリンク部分の文字色も設定しておくことをお勧めします。
なお、HTML で色を設定する方法は、さまざまな理由から非推奨とされており、新しい HTML の標準仕様では使うことができなくなっています。色を設定する場合には、できるだけスタイルシートを利用するようにしてください。

### Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>HTMLで全体の文字色を設定する</title>
</head>
<body text="#0066cc" bgcolor="#ffffff">
<h1>文字色</h1>
<p>
body要素のtext属性を使用すると、ページ全体の文字色を設定することができます。
</p>
</body>
</html>
```

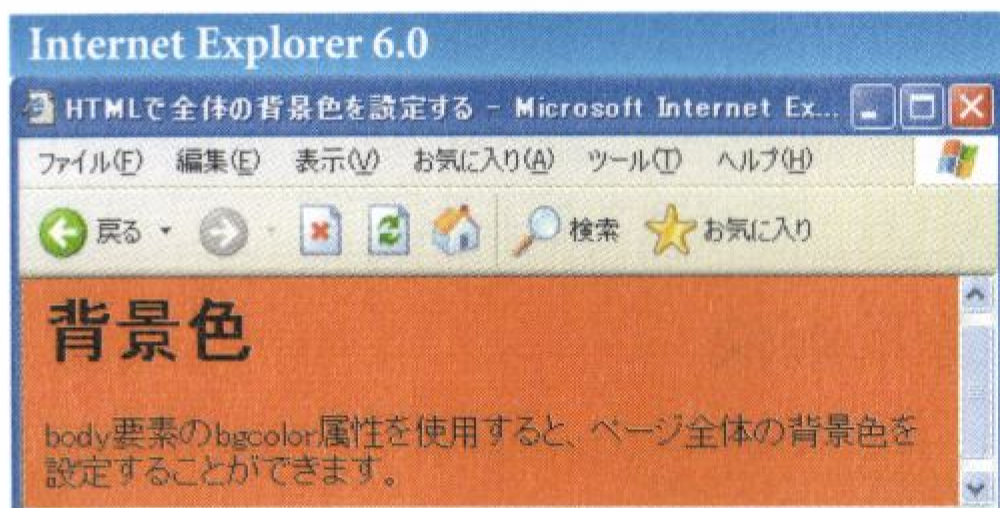

色指定：「HTMLについて」の「色の指定方法」(P.6)
<body bgcolor="～">：「基本的な内容」の「全体の背景色を設定する」(P.13)
<body link="～">：「リンク」の「リンク部分の文字色を設定する」(P.59)
スタイルシート：「フォント」の「文字色を指定する」(P.201)
色指定：巻末付録「カラーチャート1～3」(巻末)

# 全体の背景色を設定する

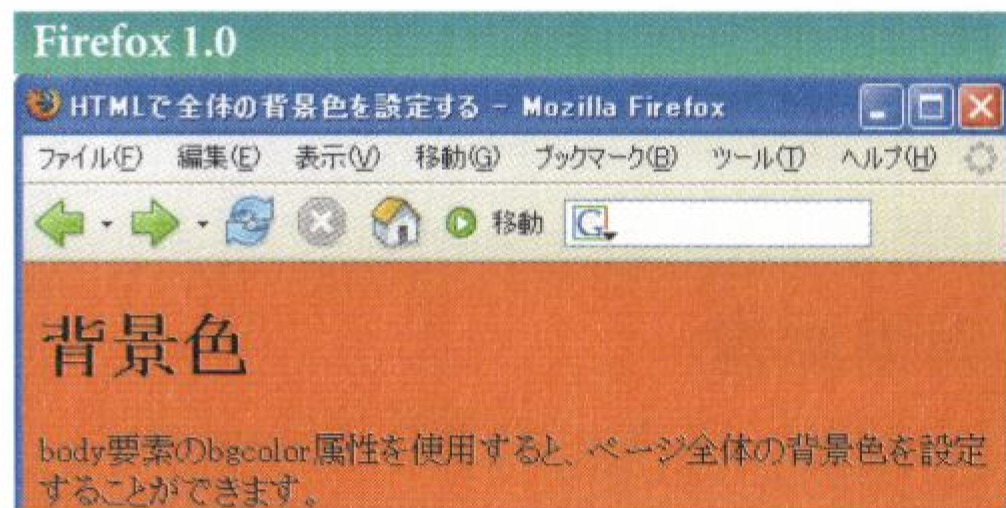
## <body bgcolor="色指定">～</body>



Internet Explorer 6.0




Firefox 1.0




body 要素の bgcolor 属性は、ページ全体の背景を、指定した色に設定します。
背景色を設定する場合には、同時に文字色も設定しておくようにしてください。そうしておかないと、ユーザーがブラウザの設定で文字色を変更している場合などに、背景色と文字色が同じ系統の色になってしまい、文字が読めなくなってしまう可能性があります。また、同様の理由で同時にリンク部分の文字色も設定しておくことをお勧めします。
なお、HTML で色を設定する方法は、さまざまな理由から非推奨とされており、新しい HTML の標準仕様では使うことができなくなっています。色を設定する場合には、できるだけスタイルシートを利用するようにしてください。

### Sample

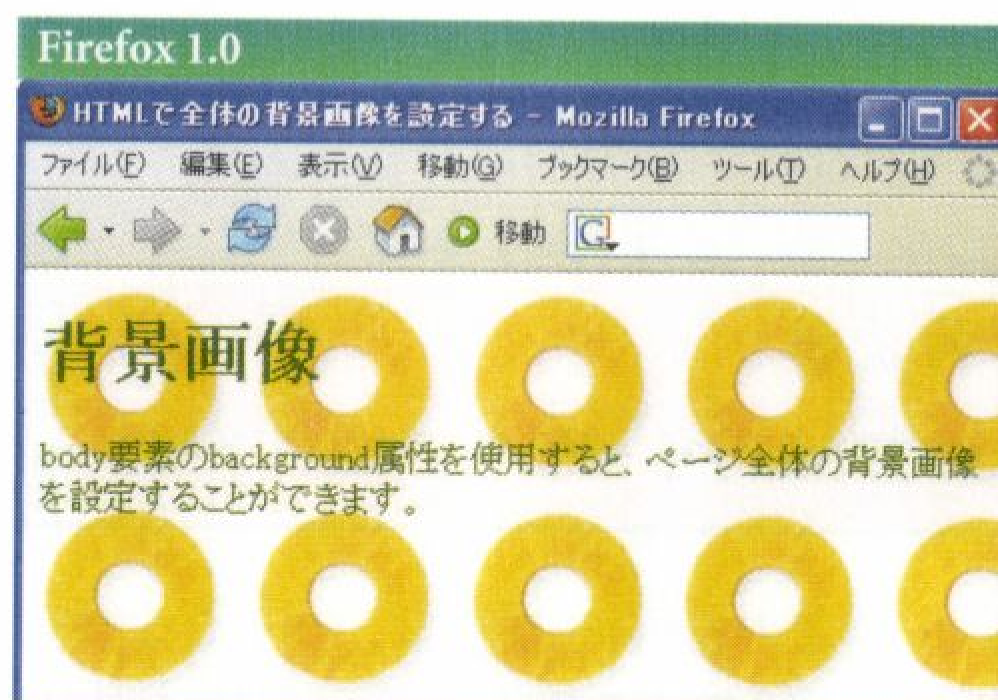
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>HTMLで全体の背景色を設定する</title>
</head>
<body text="#000000" bgcolor="#ff6600">
<h1>背景色</h1>
<p>
body要素のbgcolor属性を使用すると、ページ全体の背景色を設定することができます。
</p>
</body>
</html>
```


色指定：「HTMLについて」の「色の指定方法」(P.6)
<body text="～">：「基本的な内容」の「全体の文字色を設定する」(P.12)
<body link="～">：「リンク」の「リンク部分の文字色を設定する」(P.59)
スタイルシート：「背景」の「背景色を指定する」(P.224)
色指定：巻末付録「カラーチャート1～3」(巻末)

# 全体の背景画像を設定する

**<body background="*画像のURL*">～</body>**






背景として使用した画像

body要素のbackground属性は、ページ全体の背景に、指定した画像をタイル状に並べて表示します。
背景画像を設定する場合には、同時に文字色と背景色も設定しておくようにしてください。背景として指定した画像は、さまざまな理由によって表示されない可能性もありますし、ユーザーがブラウザの設定で文字色や背景色を変更している場合などには、文字が読みにくくなってしまう可能性もあります。また、同様の理由で同時にリンク部分の文字色も設定しておくことをお勧めします。
なお、HTMLで背景画像を設定する方法は、さまざまな理由から非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。背景画像を設定する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>HTMLで全体の背景画像を設定する</title>
</head>
<body text="#006600" background="pineapple.jpg" bgcolor="#ffffff">
<h1>背景画像</h1>
<p>
```

```
body要素のbackground属性を使用すると、ページ全体の背景画像を設定することができます。
</p>
</body>
</html>
```

スタイルシート：「背景」の「背景画像を指定する」(P.227)

## コラム

### HTML4で非推奨の要素と属性はすでに廃止されている!?

1997年の12月にW3Cの勧告として公開されたHTML4.0の仕様書では、それまで利用されていた主に表示に関する指定をするための要素や属性を「非推奨」とし、それらが将来的に廃止される予定であることを示しました。そして、それらを実際に廃止するまでの移行期間用として、HTML4.0の本来の形である「Strict」という形式の他に、非推奨の要素や属性も利用できる「Transitional」、フレームを利用できる「Frameset」というふたつの形式も用意しました。その後、HTMLはXMLによって再定義され、2000年1月にはXHTML1.0として公開されましたが、この時点でもまだ「Transitional」と「Frameset」の形式は利用できるようになっていました。しかし、2001年の5月に勧告として公開されたXHTML1.1では、ついに「Transitional」と「Frameset」という形式がなくなってしまい、結果として「Strict」と同様の形式だけがXHTML1.1ということになったのです。つまり、廃止予定とされてきた要素と属性およびフレームが、XHTML1.1でついに廃止されてしまったということになります。

※ただし、廃止された機能もモジュールという形では残されていますので、文書型を定義すれば使うことも可能となっています。

HTML

# 目的に応じて範囲を設定する

**<div>～</div>**
**<span>～</span>**

Internet Explorer 6.0




Firefox 1.0




div 要素はそれがブロックレベルの要素であることを、span 要素はそれがインラインの要素であることを示し、他には要素としての意味を特に持っていません。
これらの要素は、HTML で用意されている他の要素では示すことができないような範囲を必要に応じて設定し、その部分に対してスタイルシートを適用したり、言語の種類を示す場合などに利用されます。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
<title>目的に応じて範囲を設定する</title>
<style type="text/css">
<!--
#footer { text-align: center }
-->
</style>
</head>
<body>
  ～
<div id="footer">
<p>
<a href="index.html">ホーム</a> |
<a href="wtnew.html">更新情報</a> |
<a href="about.html">会社概要</a> |
<a href="prdct.html">製品情報</a> |
<a href="recrt.html">採用情報</a>
</p>
<p>Copyright &copy; 2001 - 2005 Div,Inc.</p>
</div>
</body>
</html>
```


スタイルシート：「CSSを適用する対象」の「IDやクラスを指定した要素に適用させる」(P.193)

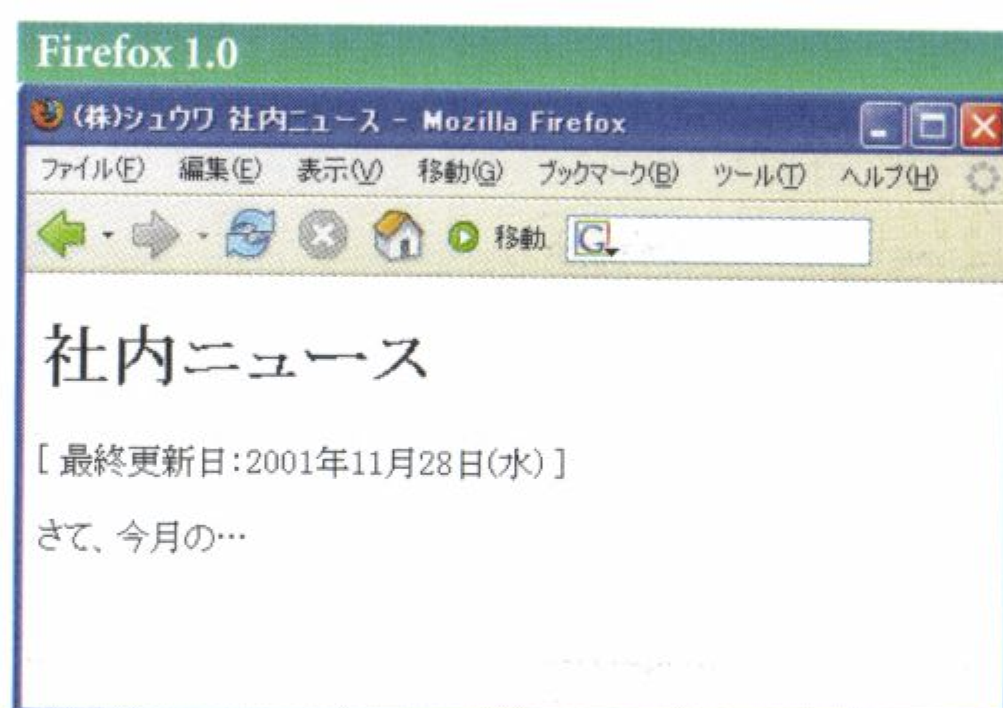
# コメントを入れる

<!--コメント文-->

Internet Explorer 6.0




Firefox 1.0




HTMLソースの中にコメントを入れておく場合に使用します。
この部分がブラウザで表示されることはありませんし、機能的にも一切影響を与えませんので、ページ更新時の注意書きなどを入れておくと便利です。コメントには任意の文字を入れることができますが、コメント文中にハイフンを連続して入れること(「--」など)は避けてください。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>(株)シュウワ 社内ニュース</title>
</head>
<body>
<h1>社内ニュース</h1>
<p>
<!--■■■更新時には、以下の行も忘れずに修正!■■■-->
[ 最終更新日:2001年11月28日(水) ]
</p>
<p>
さて、今月の…
</p>
</body>
</html>
```

HTML

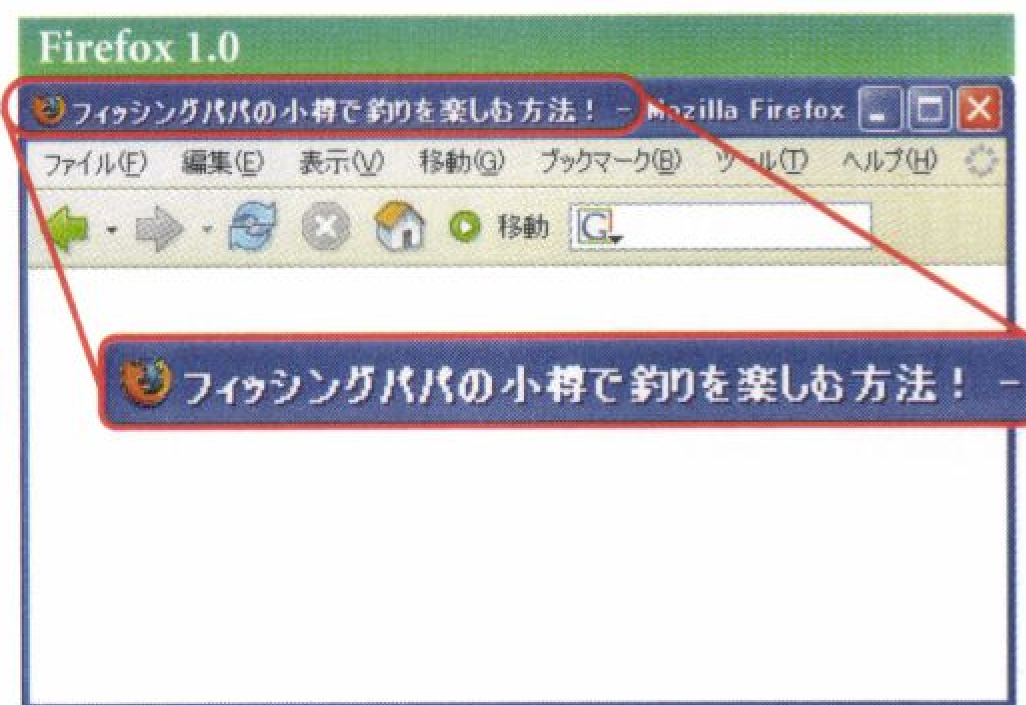
# タイトルを付ける

## <title>～</title>

Internet Explorer 6.0




Firefox 1.0




title要素は、HTML文書にタイトルを付けます。
この要素は、<head>～</head>の範囲内に必ずひとつ入れておくようにしてください。ここで指定したタイトルは、一般的なブラウザではウィンドウのタイトルバーに表示されるほか、「お気に入り」や「ブックマーク」として登録した場合のタイトルにもなります。タイトルを付ける場合には、それだけを見てページの内容が想像できるようなものにしておくとよいでしょう。たとえば、検索の結果として一覧表示された場合などに、タイトルが「はじめに」だけでは内容が想像できませんし、「会社概要」ではどこの会社の概要なのかがわからない状態となり、見てもらえる確率も低くなります。特にトップページ以外のページのタイトルを付ける場合には、注意してください。

### Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>フィッシングパパの小樽で釣りを楽しむ方法！</title>
</head>
<body>
  ∫
</body>
</html>
```

<html>・<head>・<body>：「基本的な内容」の「最低限必要な要素」(P.11)

# 文字コードを示す

**<meta http-equiv="Content-Type" content="text/html; charset= 文字コード">**

| 【文字コード】 | |
|---|---|
| Shift_JIS | シフトJIS |
| iso-2022-jp | JIS |
| EUC-JP | 日本語EUC |
| UTF-8 | UTF-8 |

meta要素のcharset属性は、そのHTML文書が、どの文字コードで書かれているかを示します。
文字コード名は、大文字で書いても小文字で書いてもかまいません。実際に作成したHTML文書の文字コードとmeta要素で指定する文字コードが違っている場合には、文字化けしてしまいますので、注意してください(Windowsなどの一般に広く使われているパソコンでは、シフトJISが採用されています)。
なお、この要素は必ず<head>～</head>の範囲内で、title要素などで日本語が現われる部分よりも前に配置してください。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>文字コードを示す</title>
  ⁞
</head>
<body>
  ⁞
</body>
</html>
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

# キーワード・内容の紹介・制作者名を入れる

**<meta name="author" content="*制作者名*">**
**<meta name="description" content="*内容の紹介*">**
**<meta name="keywords" content="*キーワード1, キーワード2, …*">**

meta要素のname属性は、content属性を使って、そのページの制作者名や内容の紹介、概要、キーワードなどを指定します。
キーワードは、半角のカンマ(,)で区切って、複数指定することができます。これらの情報は、画面上には表示されませんが、サーチエンジンが情報を収集する場合などに利用されます。特に、キーワードとして与える言葉は、そのページがうまく検索されるかどうかにも関係してきますので、慎重に考えて指定してください。
なお、これらの要素は<head>～</head>の範囲内に配置する必要があります。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<meta name="author" content="大藤　幹">
<meta name="description" content="HTMLのmeta要素に関する解説">
<meta name="keywords" content="META,meta要素,メタデータ,HTML">
<title>検索されやすいページの作り方</title>
  ∫
</head>
<body>
  ∫
</body>
</html>
```

文書情報を指定する

# スタイルシートやスクリプトの言語を示す

**<meta http-equiv="Content-Style-Type" content="スタイルシートの種類">**
**<meta http-equiv="Content-Script-Type" content="スクリプトの種類">**

| 【スタイルシートの種類】 | |
|---|---|
| text/css | CSS (Cascading Style Sheet) |
| **【スクリプトの種類】** | |
| text/javascript | JavaScript |



そのHTML文書内での、スタイルシートやスクリプトのデフォルトの言語を設定します。
style属性を使用して要素に直接スタイルシートを指定する場合や、イベント属性(イベントハンドラ)で要素に直接スクリプトを書き込むような場合には、それらの言語の種類を特定するための情報がありません。スタイルシートやスクリプトで、そのような使い方をする場合には、必ずデフォルトの言語を指定しておくようにしてください。一般的な環境では、特にデフォルトの言語を指定しなくても、スタイルシートはCSS、スクリプトはJavaScriptとして認識されるようですが、指定しておくのが正しい方法です。なお、これらの要素は<head>～</head>の範囲内に配置する必要があります。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Style-Type" content="text/css">
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>スタイルシートやスクリプトの言語を示す</title>
  ⁓
</head>
<body style="margin: 2em" onLoad="alert('読み込み完了！')">
  ⁓
</body>
</html>
```

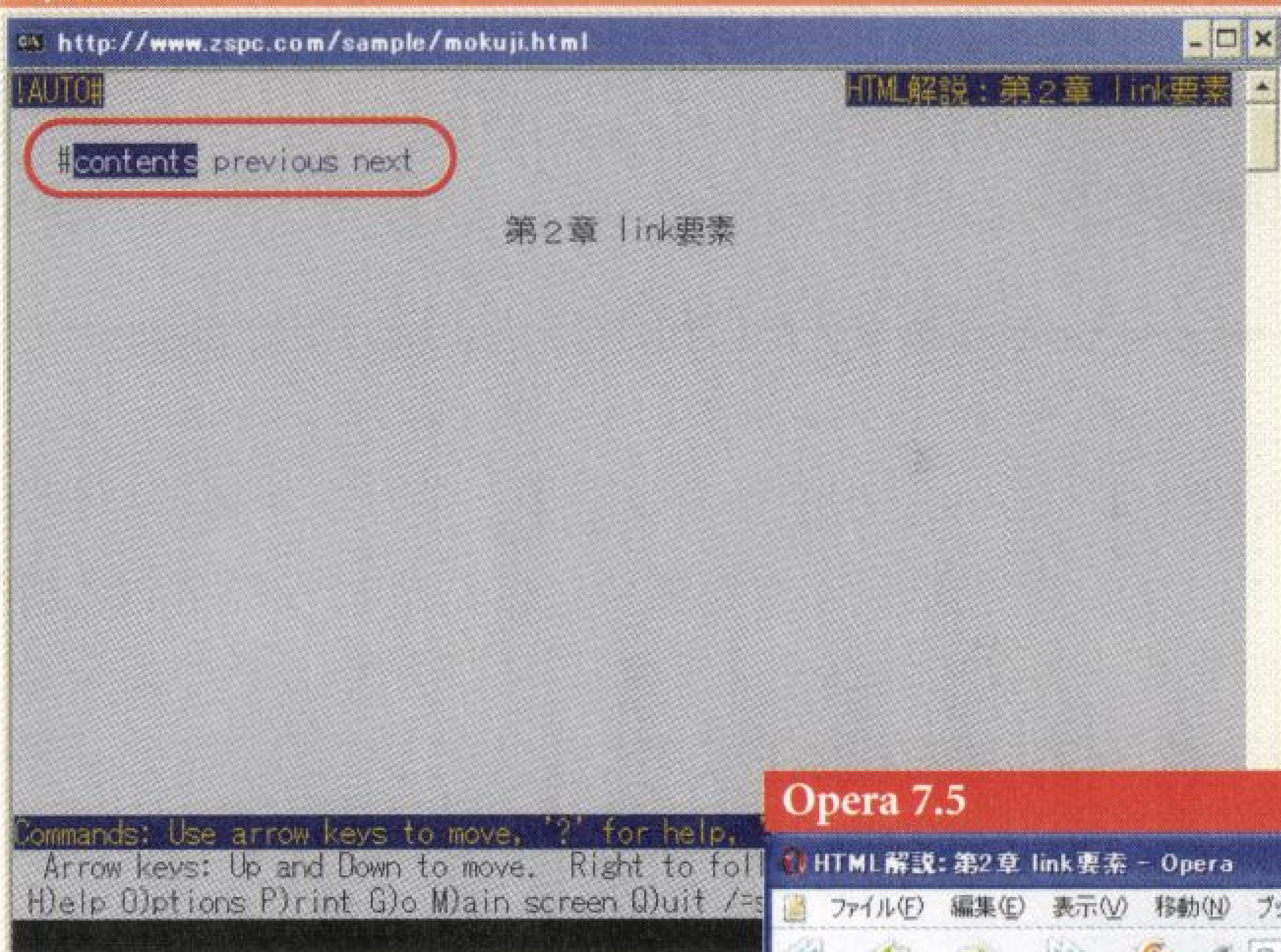

スタイルシート：「CSSの組み込み」の「任意の要素にstyle属性の値として組み込む」(P.188)
JavaScript ：「JavaScriptについて」の「イベントハンドラ」(P.306)

# 関連するほかのページを示す

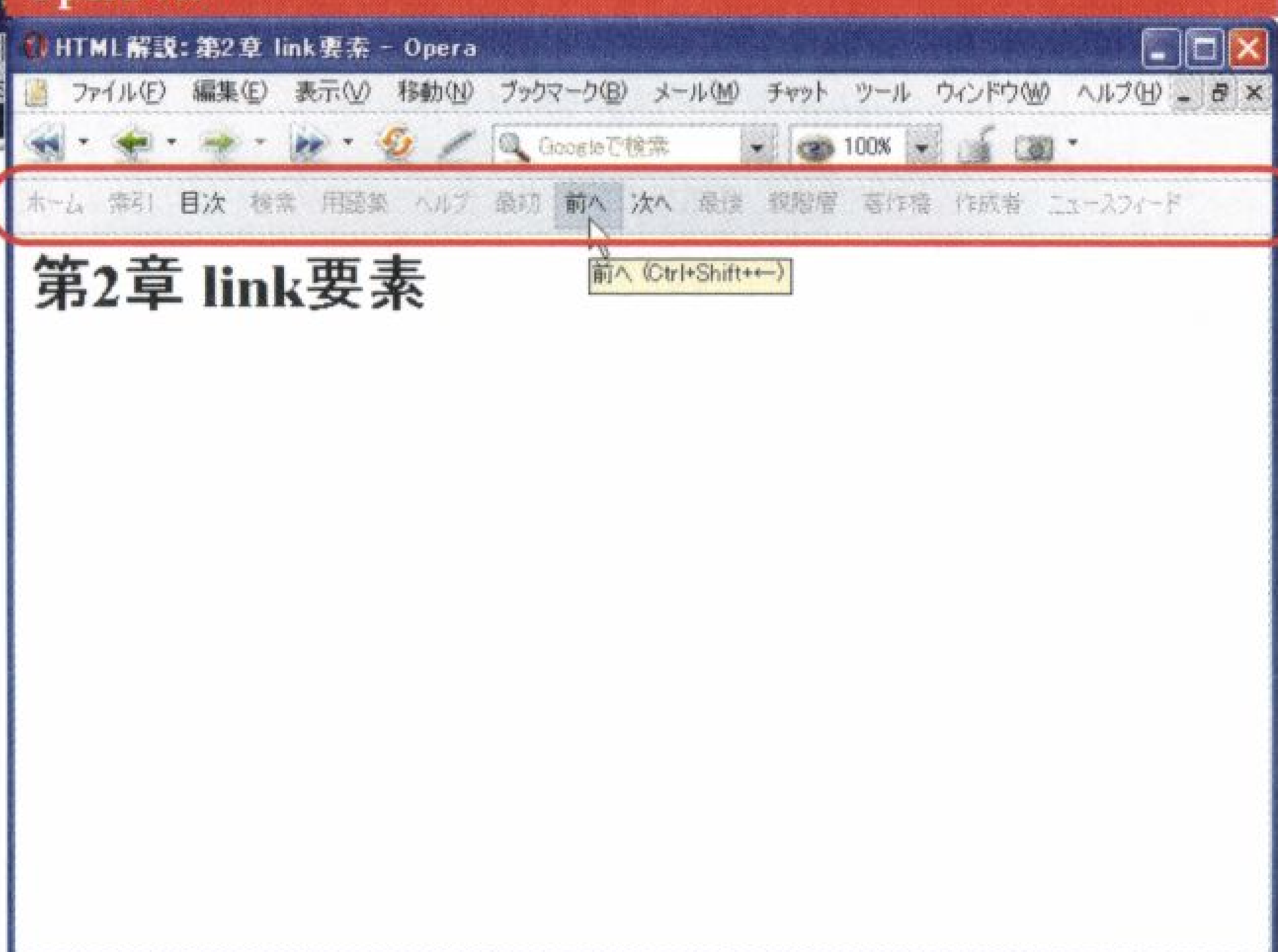
**<link rel="*関係*" href="*URL*">**
**<link rev="*関係*" href="*URL*">**

| | |
|---|---|
| rel | このページから見た、URLで示したページとの関係 |
| rev | URLで示したページから見た、このページとの関係 |

Lynx




Opera 7.5




link要素は、その文書に関連する別の文書と、その関係を示します。
たとえば、前のページや次のページ、目次のページ、外国語バージョン、別ファイルのスタイルシートなどを示す場合に使用されます。rel属性とrev属性に指定できる値については、右ページのTIPS「ページ同士の関係を表す値」を参照してください。現在のところ、一般的な多くのブラウザでは、ここで指定した情報を利用できるようにはなっていません。しかし、MozillaやOpera 7のほか、テキストブラウザのLynxではナビゲーションシステムとして利用できますし、これらの情報はサーチエンジンなどにも利用されています。
なお、link要素は必ず<head>～</head>の範囲内に配置するようにしてください。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>HTML解説：第2章 link要素</title>
<link rel="contents" href="mokuji.html">
<link rel="previous" href="chapter1.html">
<link rel="next" href="chapter3.html">
</head>
<body>
<h1>第2章 link要素</h1>
  ⁝
</body>
</html>
```

スタイルシート：「CSSの組み込み」の「CSSの書かれたファイルを読み込む」(P.183)

## TIPS

### ページ同士の関係を表す値

link要素とa要素では、rel属性とrev属性を使用して他のページとの関係を表すことができます。その関係を示す値として、HTML4.01では以下のものが定義されています（値は大文字で書いても小文字で書いてもかまいません）。

| 値 | 概要 |
|---|---|
| Alternate | 別バージョン |
| Stylesheet | 別ファイルのスタイルシート |
| Start | 最初のページ |
| Next | 次のページ |
| Prev | 前のページ |
| Contents | 目次 |
| Index | 索引 |
| Glossary | 用語集 |
| Copyright | 著作権に関するページ |
| Chapter | 章 |
| Section | 節 |
| Subsection | 項 |
| Appendix | 付録 |
| Help | ヘルプ |
| Bookmark | 同一文書内でのジャンプ先 |

# 基準URLを設定する

**<base href="*絶対URL*">**
**<base href="*絶対URL*" target="*ターゲット名*">**



base要素は、そのページで使用する相対URLの基準となる絶対URLを設定します。この指定を行うと、以降そのページで指定する相対URLは、すべてここで指定した絶対URLを基準としたものとして認識されます。この指定を行わなかった場合には、現在のページの位置が基準となります。target属性を指定すると、リンク先のページを開くデフォルトのフレームやウィンドウを指定することができます。
なお、base要素は<head>～</head>の範囲内で、相対URLを指定する他の要素よりも前に配置するようにしてください。

※下の例の場合、相対URL「../intro/url.html」は、以下の絶対URLになります。
http://www.basesample.com/intro/url.html

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" 
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<base href="http://www.basesample.com/html/index.html">
<meta http-equiv="Content-Type" content="text/html; 
 charset=Shift_JIS">
<title>基準URLを設定する</title>
</head>
<body>
<p>
基準となる<a href="../intro/url.html">絶対URL</a>を設定します。
</p>
</body>
</html>
```


絶対URL：「HTMLについて」の「ファイルの位置の指定 - 絶対URLと相対URL -」(P.6)
target属性：「リンク」の「リンク先を別のウインドウに表示する」(P.61)
TIPS「target属性の特別な4つの値」(P.62)

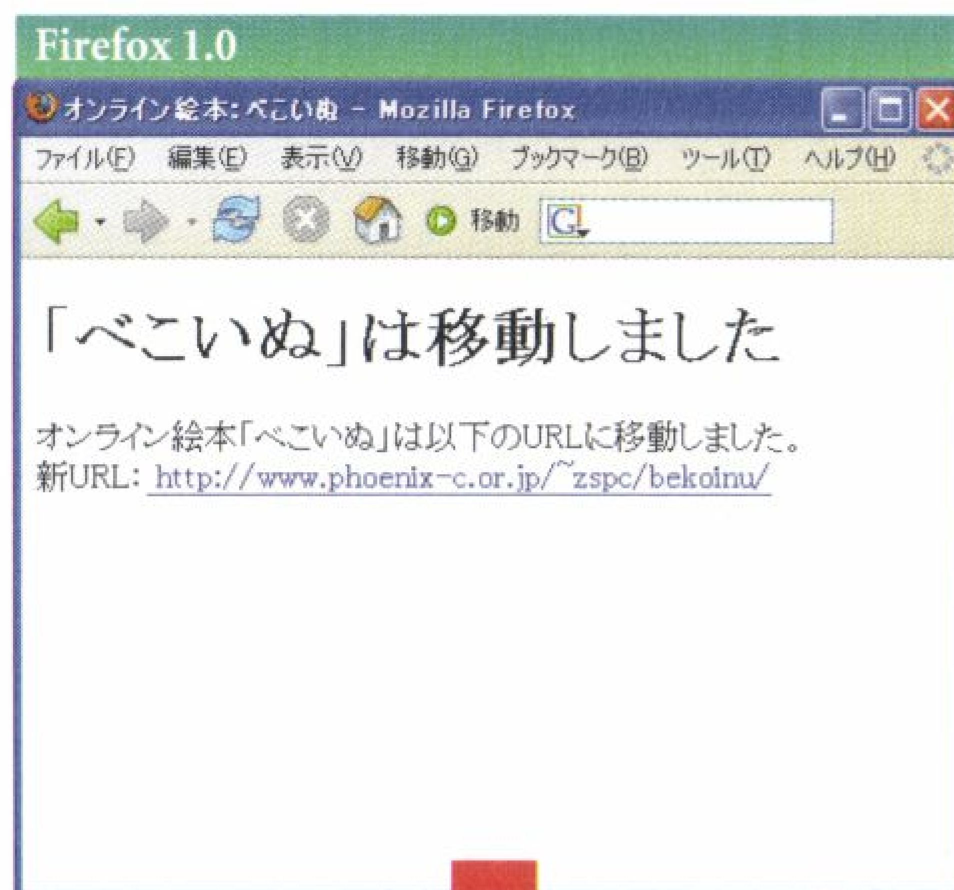
# 自動的にページを読み込む

<meta http-equiv="refresh" content="*秒数*">
<meta http-equiv="refresh" content="*秒数*;*URL*=*移動先URL*">

Internet Explorer 6.0

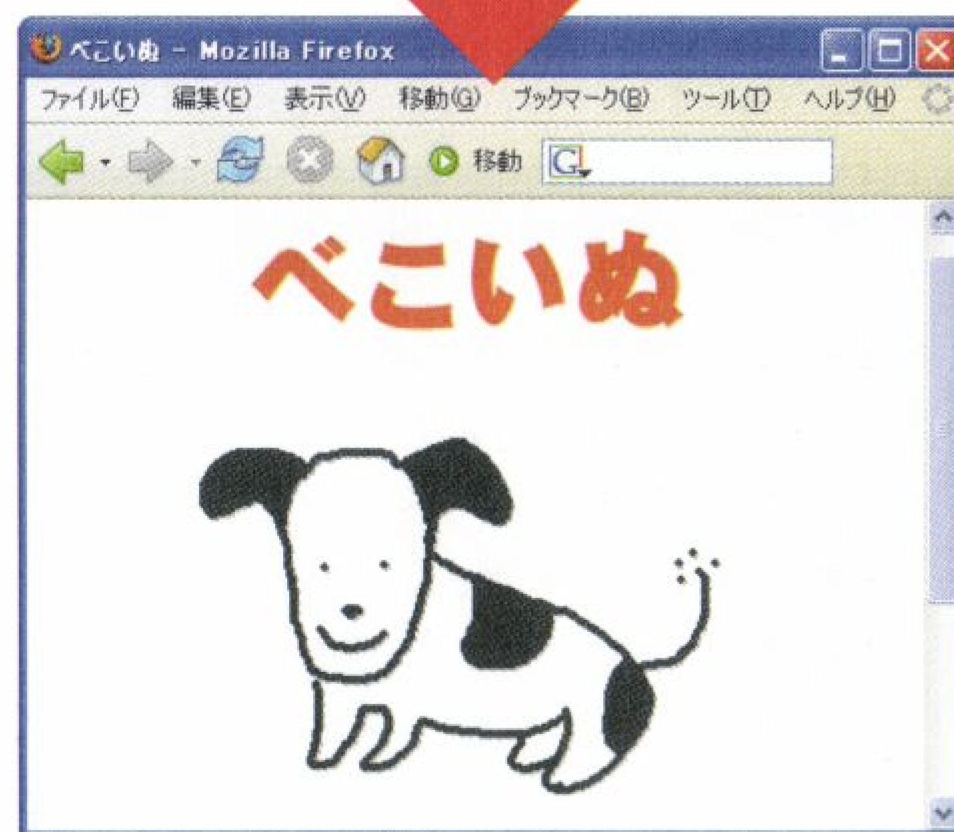



Firefox 1.0










指定した秒数後に、自動的にページの読み込みを開始します。
移動先のURLを指定した場合はそのページを読み込みますが、指定していない場合は同じページを再度読み込みます。この要素は、<head>～</head>の範囲に配置してください。この機能は、ページのURLを変更した場合などに元の古いページ(「引っ越しました」のページ)でよく利用されるものですが、すべてのブラウザでこの機能を利用できるわけではありません。自動的に別のページに移動させる場合には、そのページへのリンクも付けておくとよいでしょう。
ただし、アクセシビリティを考慮するのであれば、この機能は利用しないでください。そして、代わりにサーバー側で処理を行うか、通常のリンクでユーザーが望んだ時に移動や更新ができるようにしてください。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="refresh" content="10;URL=http://www.phoenix-c.
or.jp/~zspc/bekoinu/">
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>オンライン絵本：べこいぬ</title>
</head>
<body>
<h1>「べこいぬ」は移動しました</h1>
<p>
オンライン絵本「べこいぬ」は以下のURLに移動しました。<br>
新URL：<a href="http://www.phoenix-c.or.jp/~zspc/bekoinu/">
http://www.phoenix-c.or.jp/~zspc/bekoinu/</a>
</p>
</body>
</html>
```



## コラム

### 何語で書かれているかを示す

サンプルの中で、html要素に対して「lang="ja"」という指定があることにお気付きでしょうか。lang属性は、その要素の内容の基本となる言語を示す属性で、値の「ja」は日本語を表しています。つまり、この場合はhtml要素の内容（文書全体）が日本語であることを示しているというわけです。lang属性による言語の指定は、html要素に限らず、ほぼすべての要素に対して指定できますので、部分的に言語が変わる場合などにも使用することができます。現在のところ、この指定をしたからといって目に見えて変化があるわけではないようですが、ブラウザの種類によっては表示されるフォントなどに影響が出る場合もあります。

また、翻訳サービス（ソフト）やサーチエンジンなど、言語の特定が必要となるものに対しても明確な情報を与えることができますので、lang属性は常に指定しておくようにするとよいでしょう。lang属性の値として指定できる主な言語コードは、以下の通りです。

**【主な言語コード】**

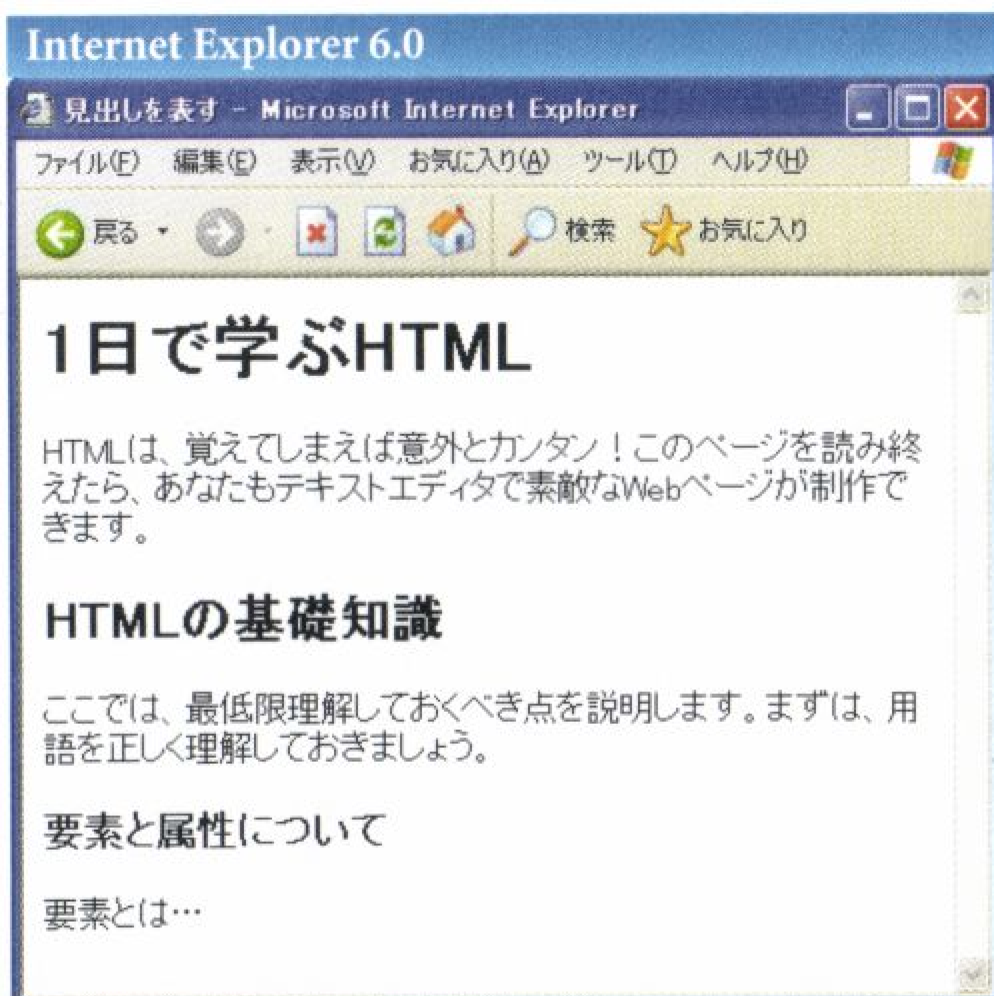
| | | | |
|---|---|---|---|
| 日本語 | ja | 韓国語 | ko |
| 中国語 | zh | 英語 | en |
| アメリカ英語 | en-US | フランス語 | fr |
| ドイツ語 | de | イタリア語 | it |
| スペイン語 | es | ロシア語 | ru |

# 見出しを表す

<h1>～</h1>
<h2>～</h2>
<h3>～</h3>
<h4>～</h4>
<h5>～</h5>
<h6>～</h6>



Internet Explorer 6.0




Firefox 1.0




hX要素は、その部分が見出し(heading)であることを示します。
1～6の数字は見出しのレベルを表しており、<h1>が1番上のレベルの大見出し、<h6>が1番下のレベルの小見出しというように6段階まで用意されています。見出しの階層に応じて、それに対応するレベルのものを使用するようにしてください。一般的なブラウザでは、上のレベルの見出しほど大きな文字で太字で表示されます。
たとえ見出しとして画像を使う場合でも、このタグで囲うようにしてください。そうすれば画像を表示できない環境でも、img要素のalt属性で指定した文字が見出しとして適切に機能します。



## Sample

```
<body>
<h1>1日で学ぶHTML</h1>
<p>
HTMLは、覚えてしまえば意外とカンタン！このページ
を読み終えたら、あなたもテキストエディタで素敵な
Webページが制作できます。
</p>
<h2>HTMLの基礎知識</h2>
<p>
```

```
ここでは、最低限理解しておくべき点を説明します。
まずは、用語を正しく理解しておきましょう。
</p>
<h3>要素と属性について</h3>
<p>
要素とは…
</p>
</body>
```

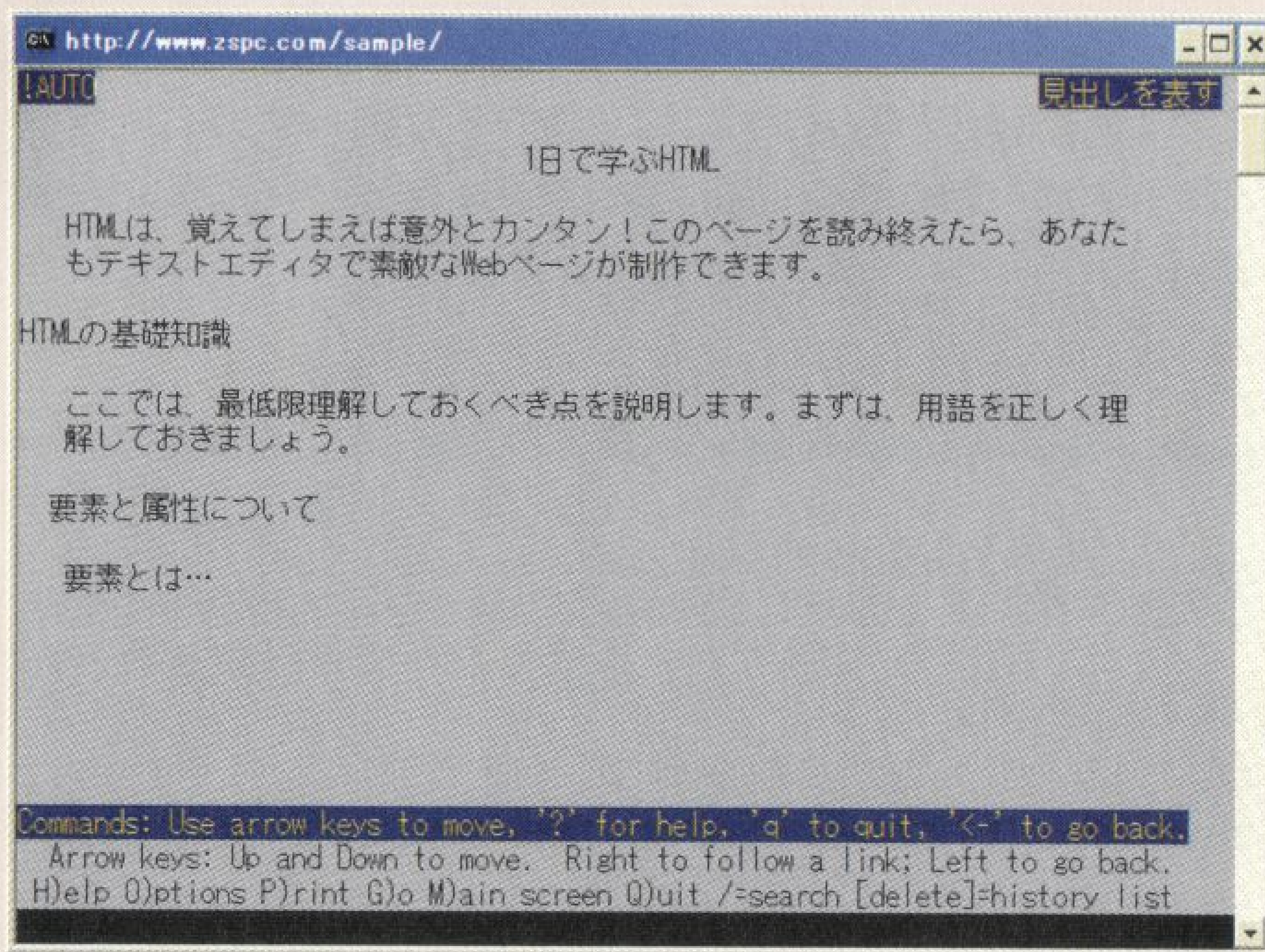
<img>：「画像とマルチメディア」の「画像を配置する」(P.99)

## コラム

### 見出しがどう表示されるかは決まっているわけではない！？

見出しのレベルの違い（h1～h6）は、必ずしも文字の大きさで表現されるわけではありません。たとえば、テキストブラウザのLynxでは、上のサンプルを表示させると下の図のようになります。見出しのレベルを大きさではなく、表示位置で示しているわけです。他にも、音声ブラウザのホームページ・リーダー3.01では、見出しの前に特定の音を出して、しかもゆっくりと読み上げるなど、ブラウザや環境によって見出しの表現方法はさまざまです。

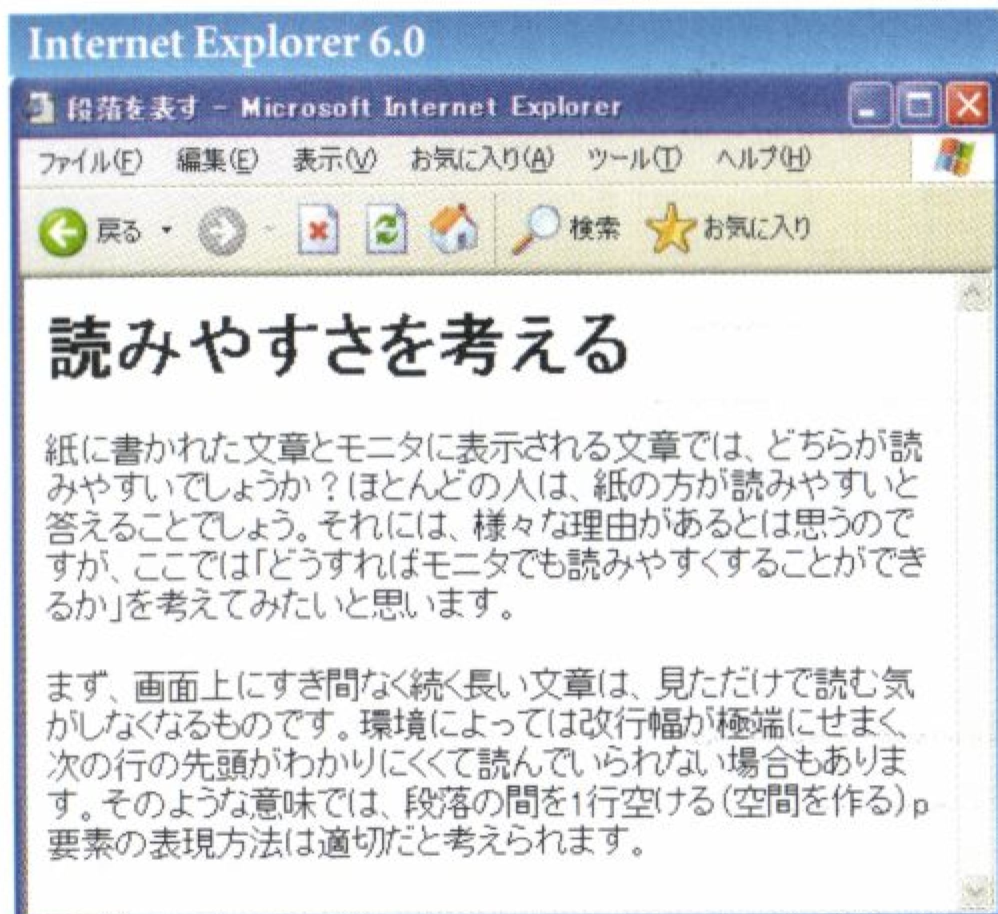
font要素を使用して単純に文字サイズを大きくすることなどで見出しを表現していると、環境によっては見出しとして認識することができなくなってしまいますので、注意してください。



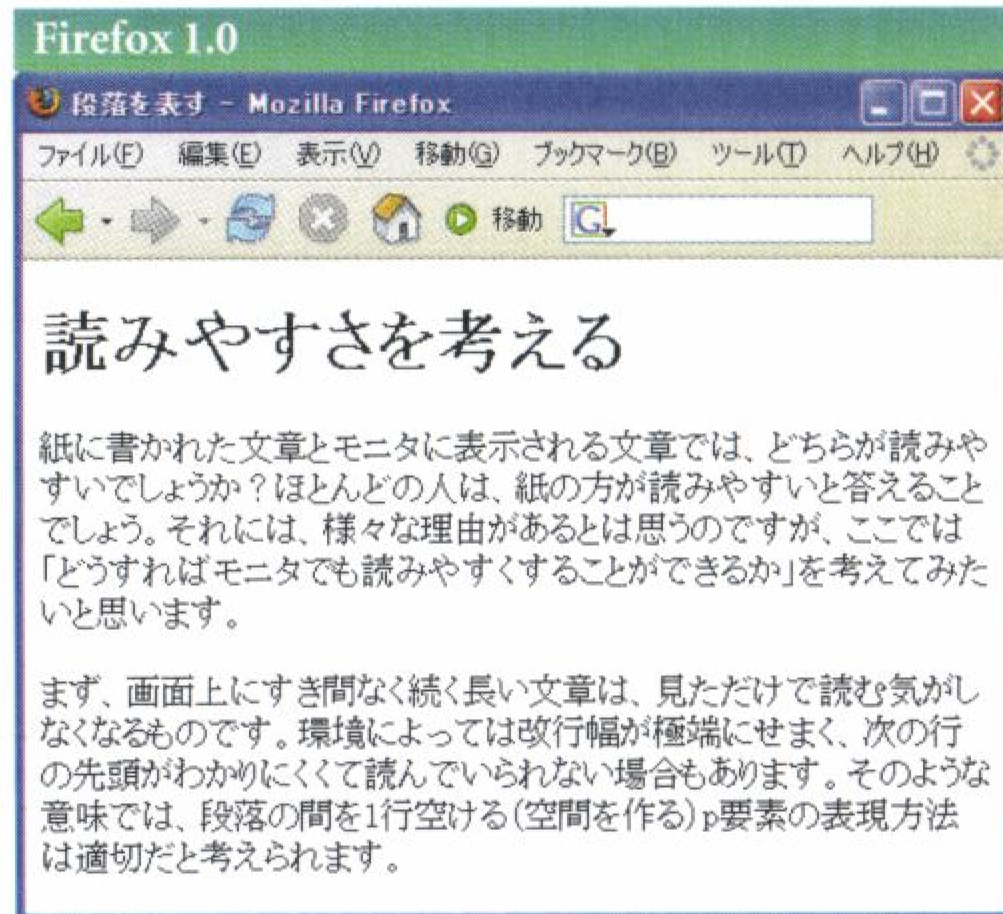
# 段落を表す

## <p>～</p>

Internet Explorer 6.0




Firefox 1.0




p要素は、その部分がひとつの段落(paragraph)であることを示します。
一般的なブラウザでは、この要素の前後に1行分のスペースがとられて表示されます。
この要素の内容としては、ブロックレベルの要素を含むことができませんので、注意してください。

### Sample

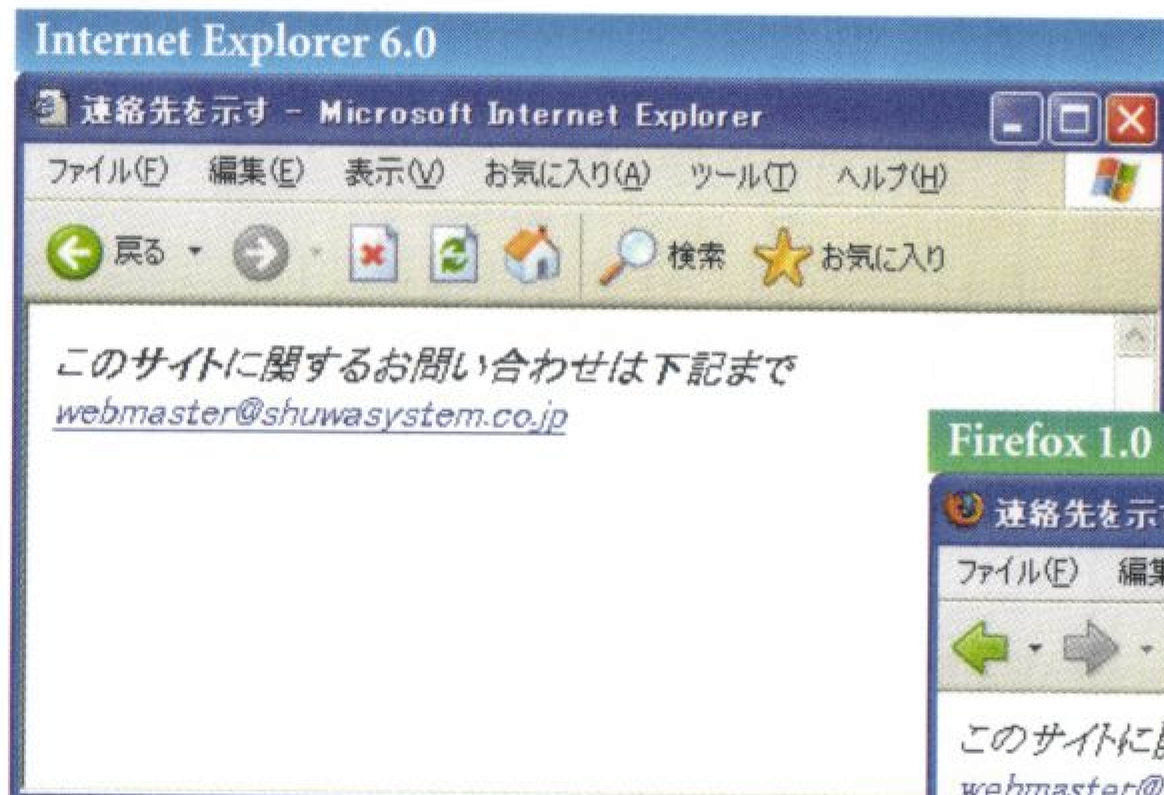
```
<body>
<h1>読みやすさを考える</h1>
<p>
紙に書かれた文章とモニタに表示される文章では、どちらが読みやすいでしょうか？ほとんどの人は、紙の方が読みやすいと答えることでしょう。それには、さまざまな理由があるとは思うのですが、ここでは「どうすればモニタでも読みやすくすることができるか」を考えてみたいと思います。
</p>
<p>
まず、画面上にすき間なく続く長い文章は、見ただけで読む気がしなくなるものです。環境によっては改行幅が極端にせまく、次の行の先頭がわかりにくくて読んでいられない場合もあります。そのような意味では、段落の間を1行空ける(空間を作る)p要素の表現方法は適切だと考えられます。
</p>
  〜
</body>
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac
i-mode
テキストの種類を指定する

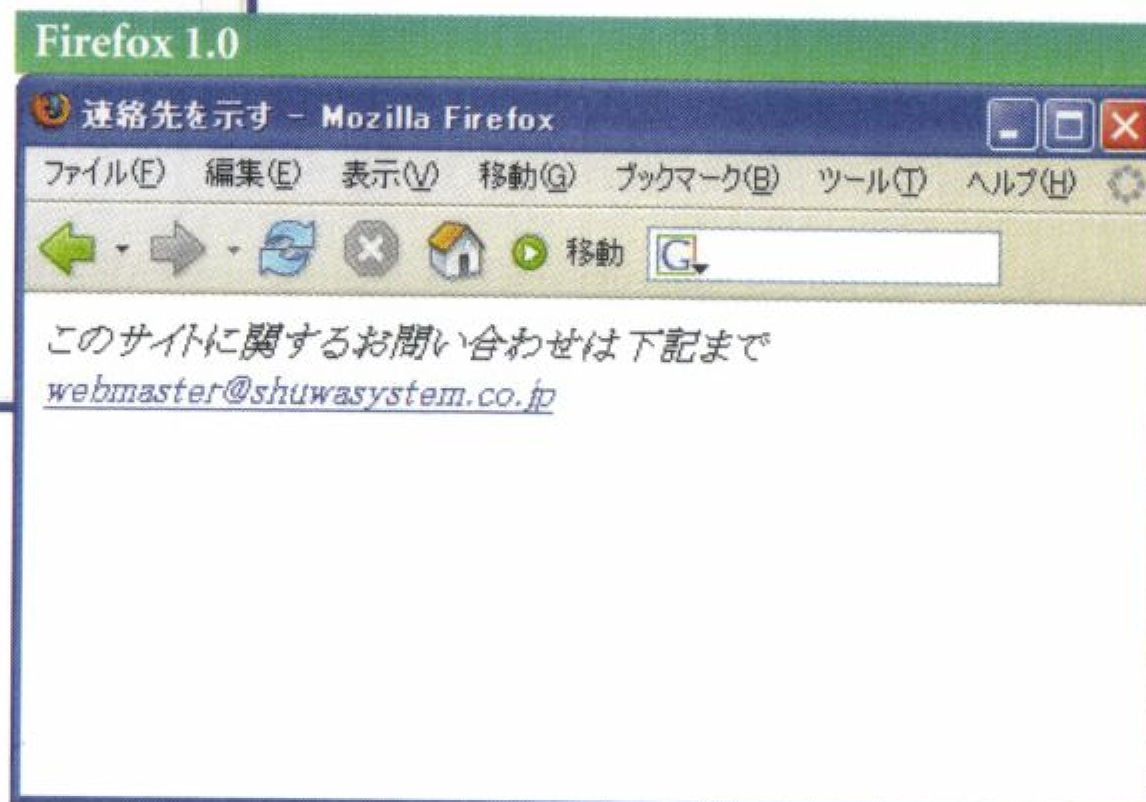
# 連絡先を示す

## <address>～</address>

Internet Explorer 6.0




Firefox 1.0




address要素は、そのページの制作者の連絡先や内容に関する問い合わせ先が書かれている部分であることを示します。
具体的には、電子メールアドレス・制作者・担当者名・住所・電話番号・FAX番号などを示す場合に使用します。一般的なブラウザではイタリックで表示されますが、スタイルシートの指定(font-style: normal)によって標準の字体に戻すことも可能です。
なお、この要素の内容としては、ブロックレベルの要素を含むことができませんので、注意してください。

### Sample

```
<body>
  ≀
<address>
このサイトに関するお問い合わせは下記まで<br>
<a href="mailto:webmaster@shuwasystem.co.jp">↵
webmaster@shuwasystem.co.jp</a>
</address>
</body>
```

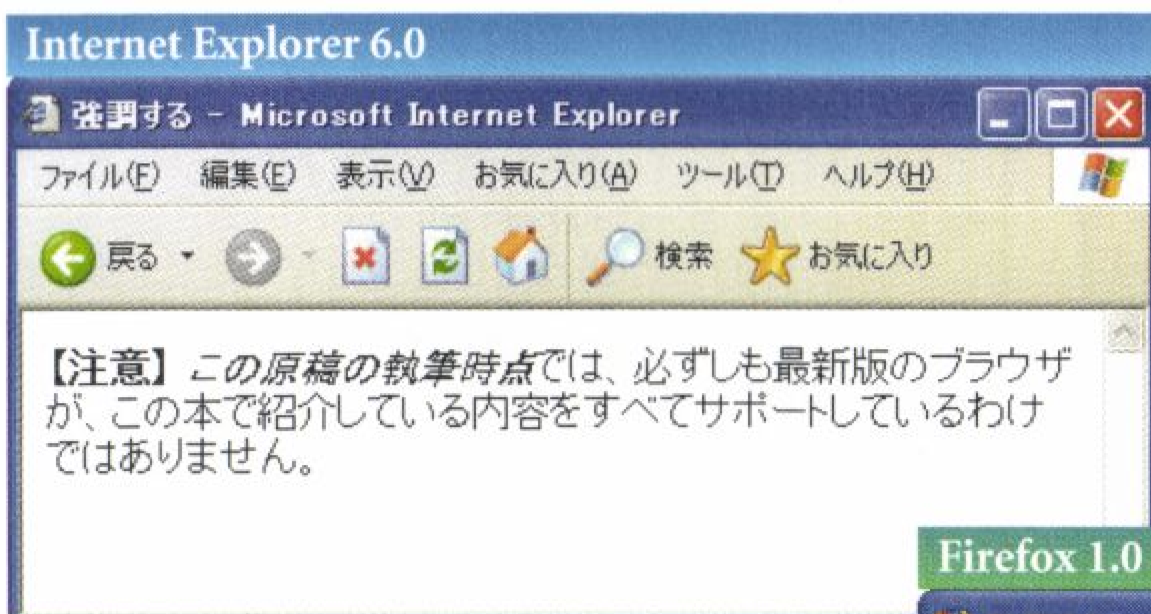
スタイルシート:「フォント」の「フォントスタイルを指定する」(P.208)

# 強調する

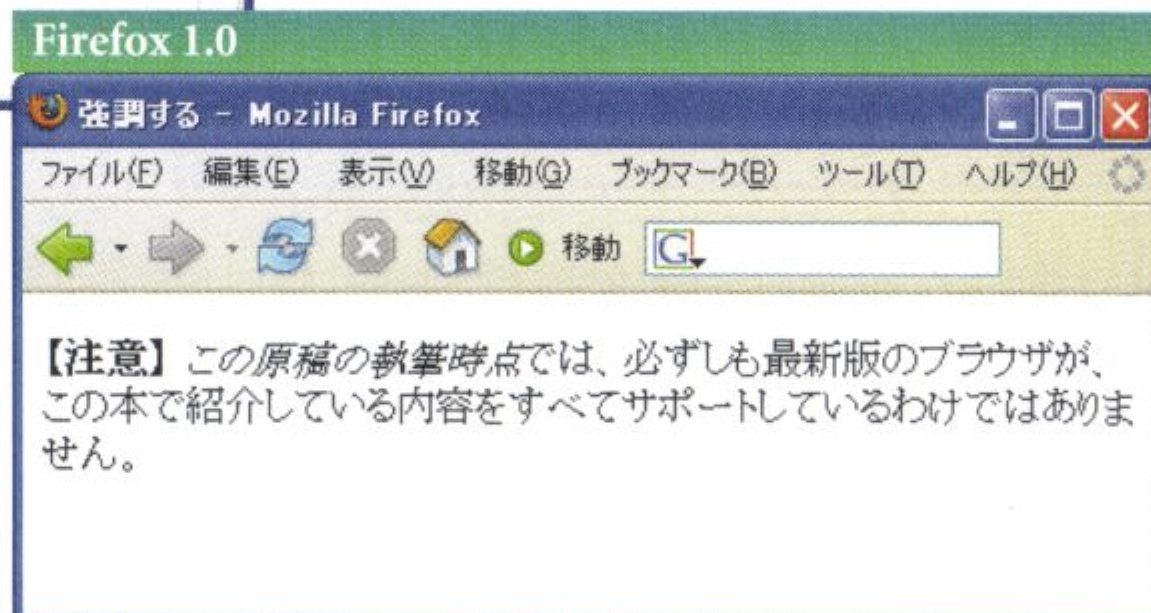
**<em>～</em>**
**<strong>～</strong>**



Internet Explorer 6.0




Firefox 1.0




その部分が、強調されていることを示します。
em要素は普通の強調を、strong要素はより強い強調であることを示します。一般的なブラウザでは、em要素はイタリックで、strong要素は太字で表示されます。

## Sample

```
<p>
<strong>【注意】</strong>
<em>この原稿の執筆時点</em>では、必ずしも最新版のブラウザが、
この本で紹介している内容をすべてサポートしているわけではありません。
</p>
```

### コラム

#### q要素の対応状況について

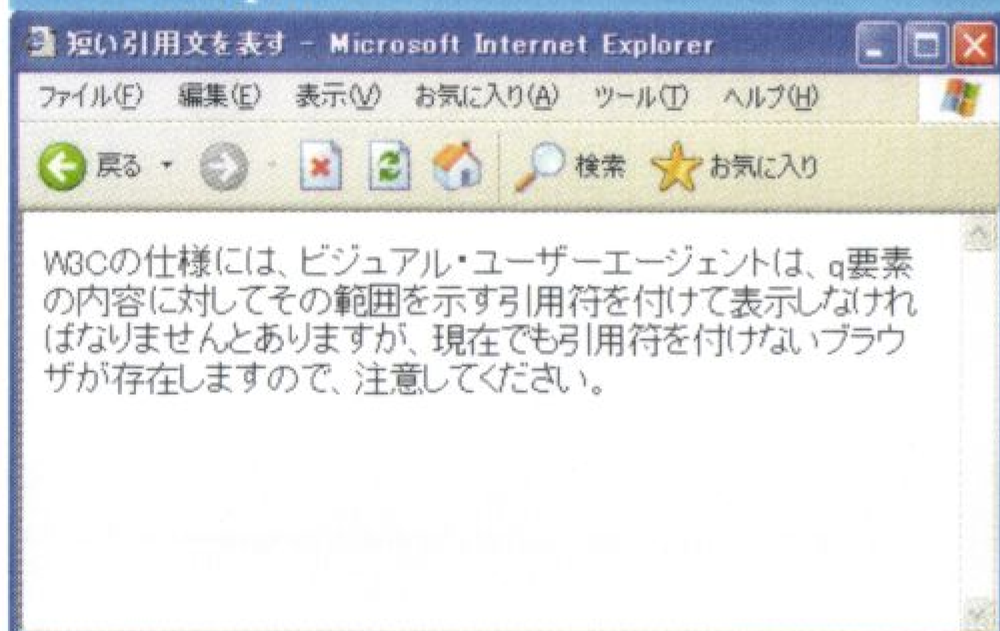
Netscape 6以降では、q要素の前後に引用符を付けて表示するだけでなく、cite属性で指定されたURLに移動することもできるようになっています。また、OperaやNetscapeのバージョン6以降、Firefox、Mozillaでは、スタイルシートを利用することで、引用符として使用する記号を任意のものに変更することもできます。他に、Macintosh版のInternet Explorer 5やSafariなどでも引用符を表示しますが、Windows版のInternet Explorerはバージョン6.0でも引用符を表示しません。

HTML

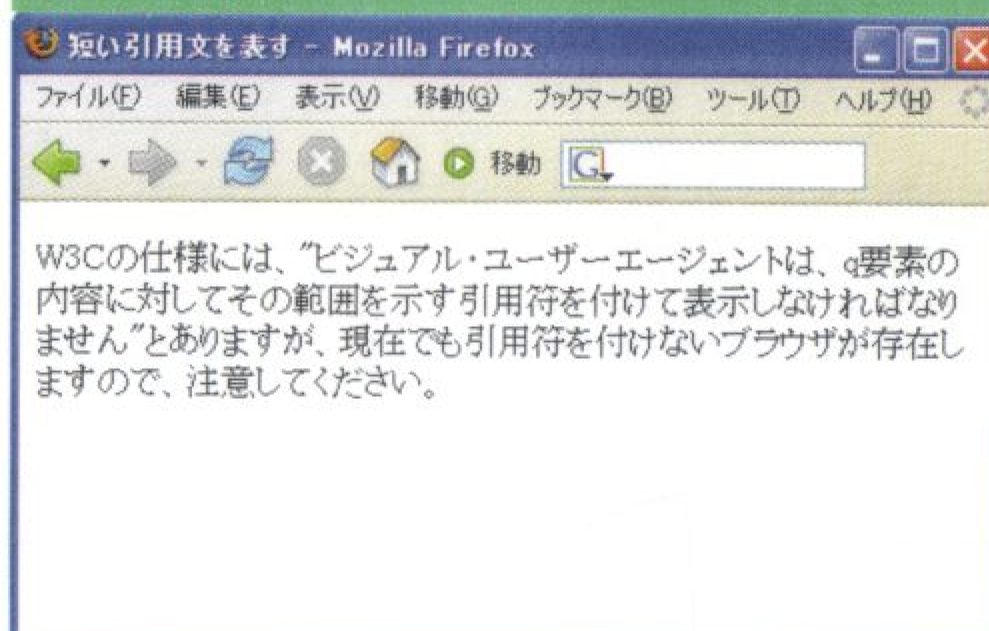
# 短い引用文を表す

<q>～</q>
<q cite="*引用元のURL*">～</q>

Internet Explorer 6.0




Firefox 1.0




q要素は、その部分が、短い引用文であることを示します。
段落の一部など(インライン要素)として引用する場合に使用されます。cite属性を使用すると、引用したページのURLを示すこともできます。この要素がサポートされているブラウザでは、指定した範囲の前後に自動的に引用符が付けられますので、文章の中には引用符を入れないようにしてください。

## Sample

```
<p>
W3Cの仕様には、<q cite="http://www.w3.org/TR/html401">
ビジュアル·ユーザーエージェントは、q要素の内容に対して
その範囲を示す引用符を付けて表示しなければなりません</q>
とありますが、現在でも引用符を付けないブラウザが存在
しますので、注意してください。
</p>
```

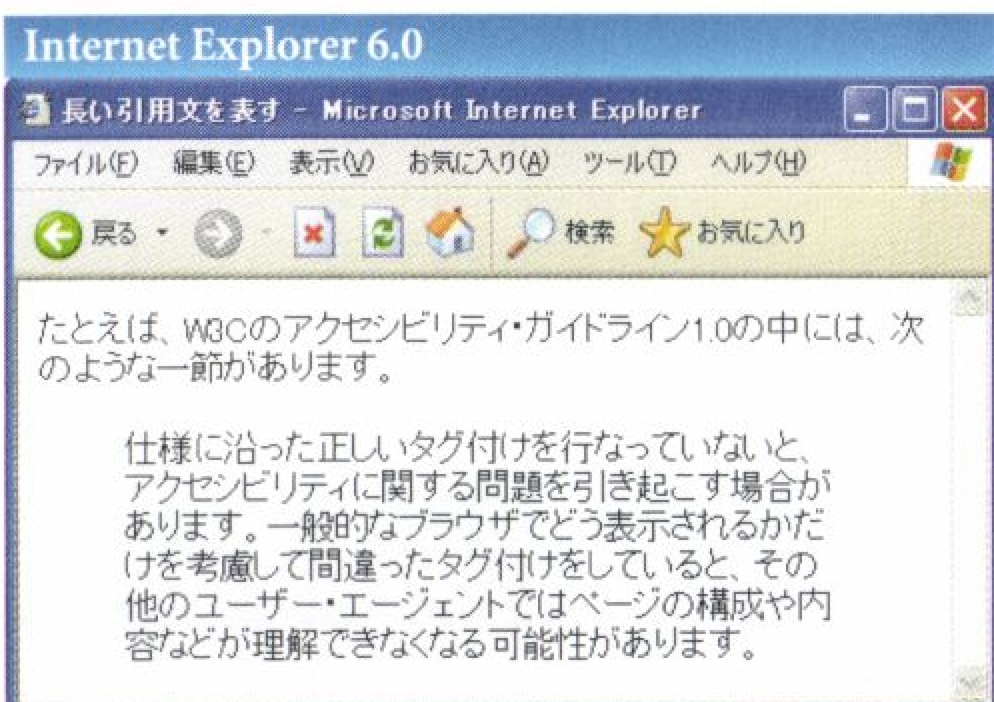

コラム「q要素の対応状況について」(P.31)
スタイルシート:「その他」の「引用符として使用する記号を設定する」(P.283)

# 長い引用文を表す

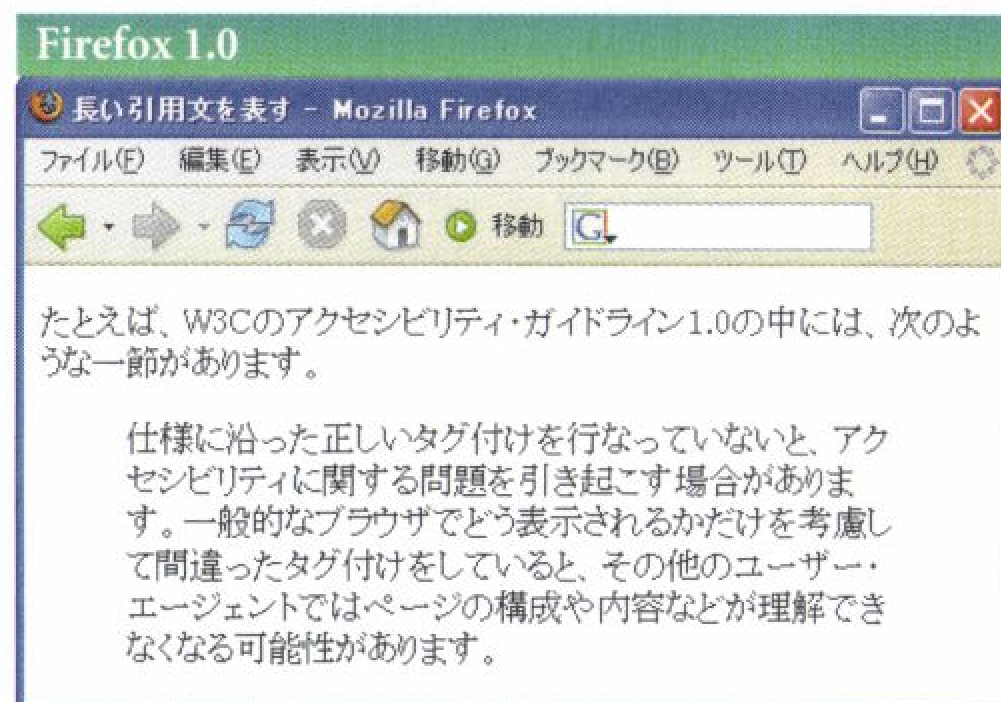
**<blockquote>～</blockquote>**
**<blockquote cite="*引用元のURL*">～</blockquote>**



Internet Explorer 6.0




Firefox 1.0




blockquote要素は、その部分が、長い引用文であることを示します。
文章を段落などのまとまった単位で(ブロックレベル要素として)引用する場合に使用されます。cite属性を使用すると、引用したページのURLを示すこともできます。この要素は、一般的なブラウザでは左右がインデントされた状態で表示されますので、かつては左右のマージンをとる目的だけで利用されることもありました。しかし、それではその部分全体が引用された文章であることになってしまいますので、マージンを設定したい場合にはスタイルシートを使うようにしてください。

## Sample

```
<p>
たとえば、W3Cのアクセシビリティ・ガイドライン1.0の中には、次のような一節があります。
</p>
<blockquote cite="http://www.w3.org/TR/WAI-WEBCONTENT/">
<p>
仕様に沿った正しいタグ付けを行なっていないと、アクセシビリティに関する問題を引き起こす場合
があります。一般的なブラウザでどう表示されるかだけを考慮して間違ったタグ付けをしていると、
その他のユーザー・エージェントではページの構成や内容などが理解できなくなる可能性があります。
</p>
</blockquote>
```


スタイルシート：「ボックス」の「マージンを設定する」(P.237)

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

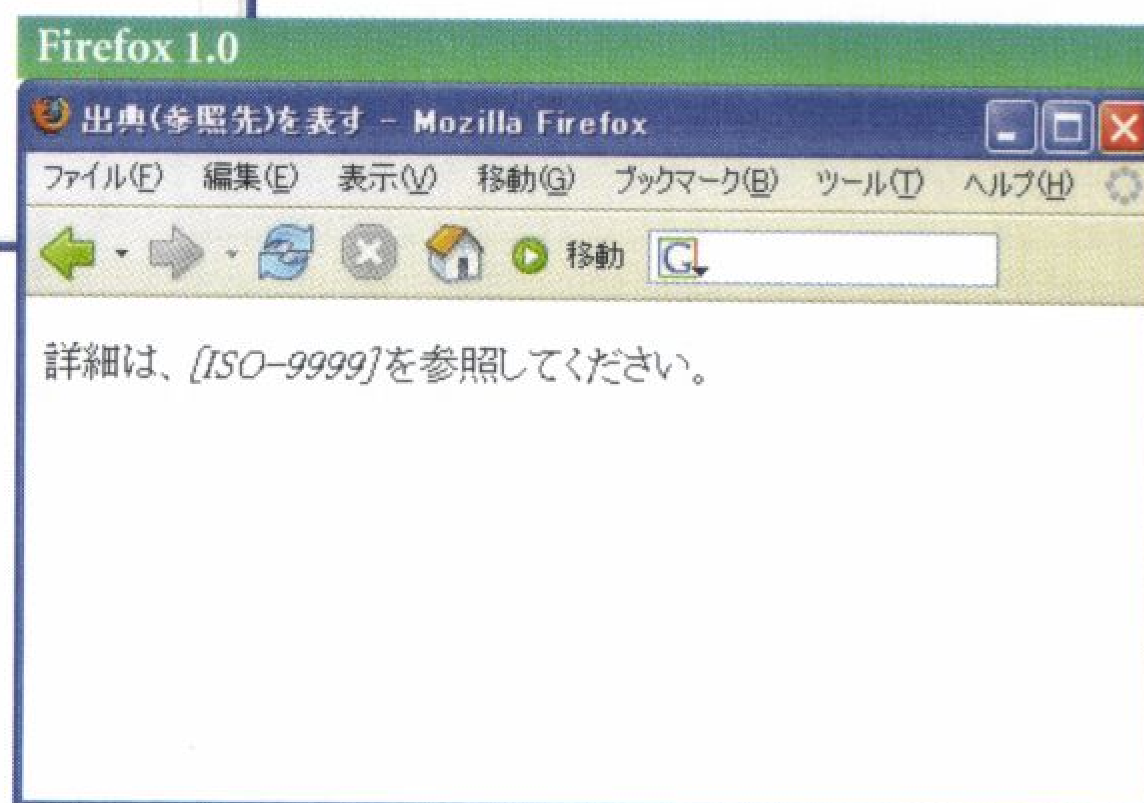
# 出典（参照先）を表す

## <cite>～</cite>

Internet Explorer 6.0




Firefox 1.0




cite要素は、その部分が、出典や参照先であることを示します。
文章を引用するのではなく、書籍や文書、規格、著者などの名前やタイトルを示す場合に使用します。一般的なブラウザでは、イタリックで表示されます。

### Sample

```
<p>
詳細は、<cite>[ISO-9999]</cite> を参照してください。
</p>
```

スタイルシート：「フォント」の「フォントスタイルを指定する」(P.208)

# 略語を表す

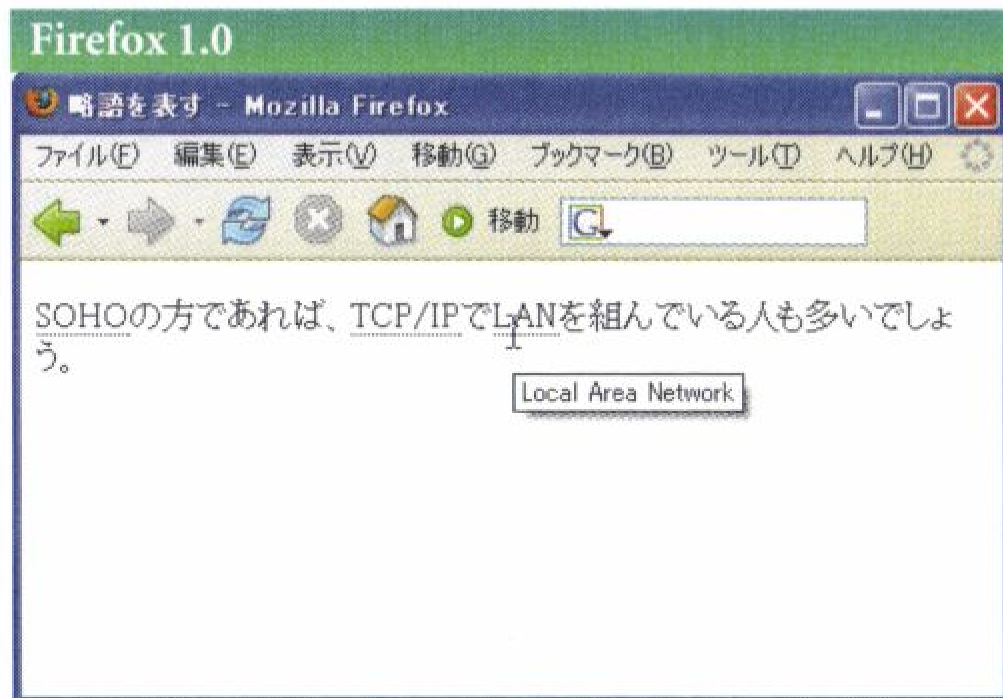
```
<abbr title="文字列">～</abbr>          ←略語
<acronym title="文字列">～</acronym>    ←頭字語
```

| | |
|---|---|
| title | 省略していない状態の言葉(文字列) |

Internet Explorer 6.0




Firefox 1.0




その部分が、略語であることを示します。

abbr要素は略語全般を示し、acronym要素は略語の中でも単語の先頭の文字を組み合わせて作られたもので、しかもひとつの単語として発音されるもの(NATO、SOHO、LANなど)を示します。title属性には、省略しない状態の言葉を指定します。

## Sample

```
<p>
<acronym title="Small Office Home Office">SOHO</acronym>
の方であれば、<abbr title="Transmission Control Protocol/
Internet Protocol">TCP/IP</abbr>で<acronym title="Local
Area Network">LAN</acronym>を組んでいる人も多いでしょう。
</p>
```

## TIPS

### 追加や削除をした日時の表し方

ins要素とdel要素で指定する日時のフォーマットは、ISO8601に準拠した形式で一見しただけでは複雑そうですが、基本的には年月日時分秒とタイムゾーンを示すだけです。

以下に示す書式で書けばよいのですが、年は4桁、その他は2桁で固定となっていますので、注意してください。

Timeの「T」で固定　日本時間の場合はこのまま

**2001-04-19T23:14:00+09:00**

年　月　日　時　分　秒

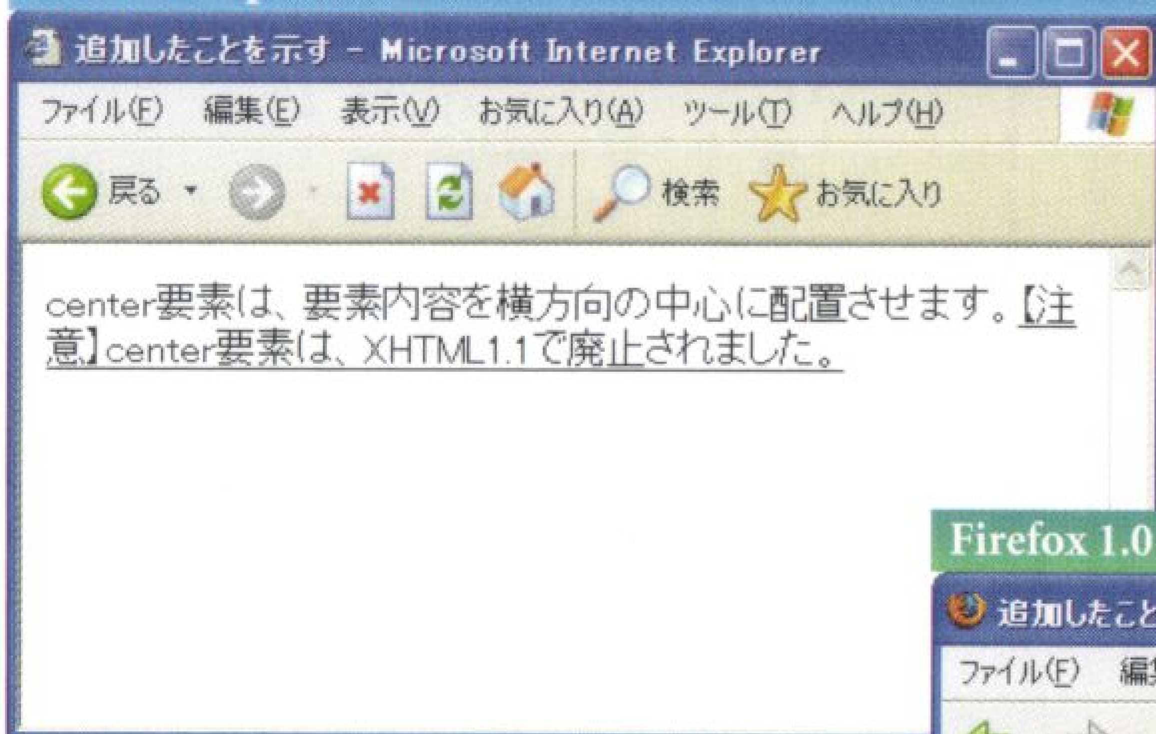
IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 追加したことを示す

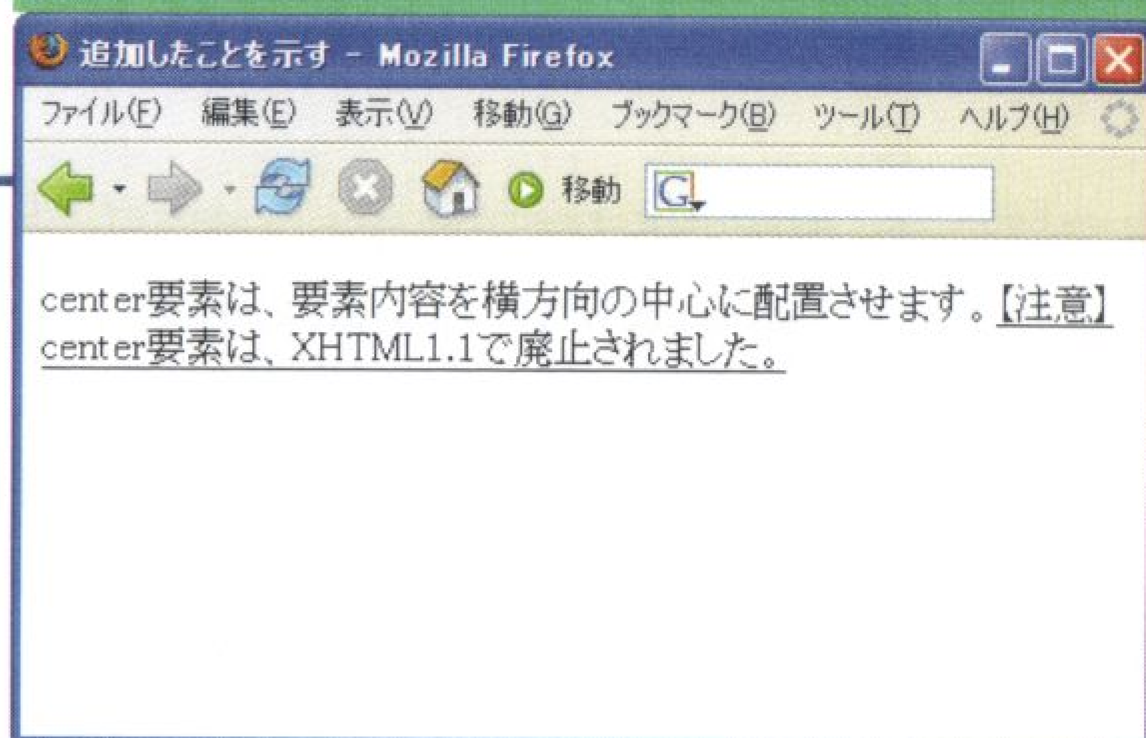
**<ins cite="*URL*" datetime="*追加日時*">～</ins>**

| | |
|---|---|
| cite | 追加に関する情報が書かれている文書のURL |
| datetime | 追加した日時（ISO8601形式） |

Internet Explorer 6.0




Firefox 1.0




ins要素は、その部分が、後から追加した部分であることを示します。
ブラウザによって表示方法は異なりますが、一般的には下線の付いた状態やイタリックなどで表示されます。また、ブラウザによっては、title属性で指定した短い説明をツールチップとして表示させることもできます。この要素は、指定する範囲に応じて、インライン要素としてもブロックレベル要素としても使用することができます。

## Sample

```
<p>
center要素は、要素内容を横方向の中心に配置させます。
<ins cite="http://www.w3.org/TR/xhtml11" datetime="2001-05-
31T22:14:00+09:00">
【注意】center要素は、XHTML1.1で廃止されました。
</ins>
</p>
```


TIPS「追加や削除をした日時の表し方」(P.35)

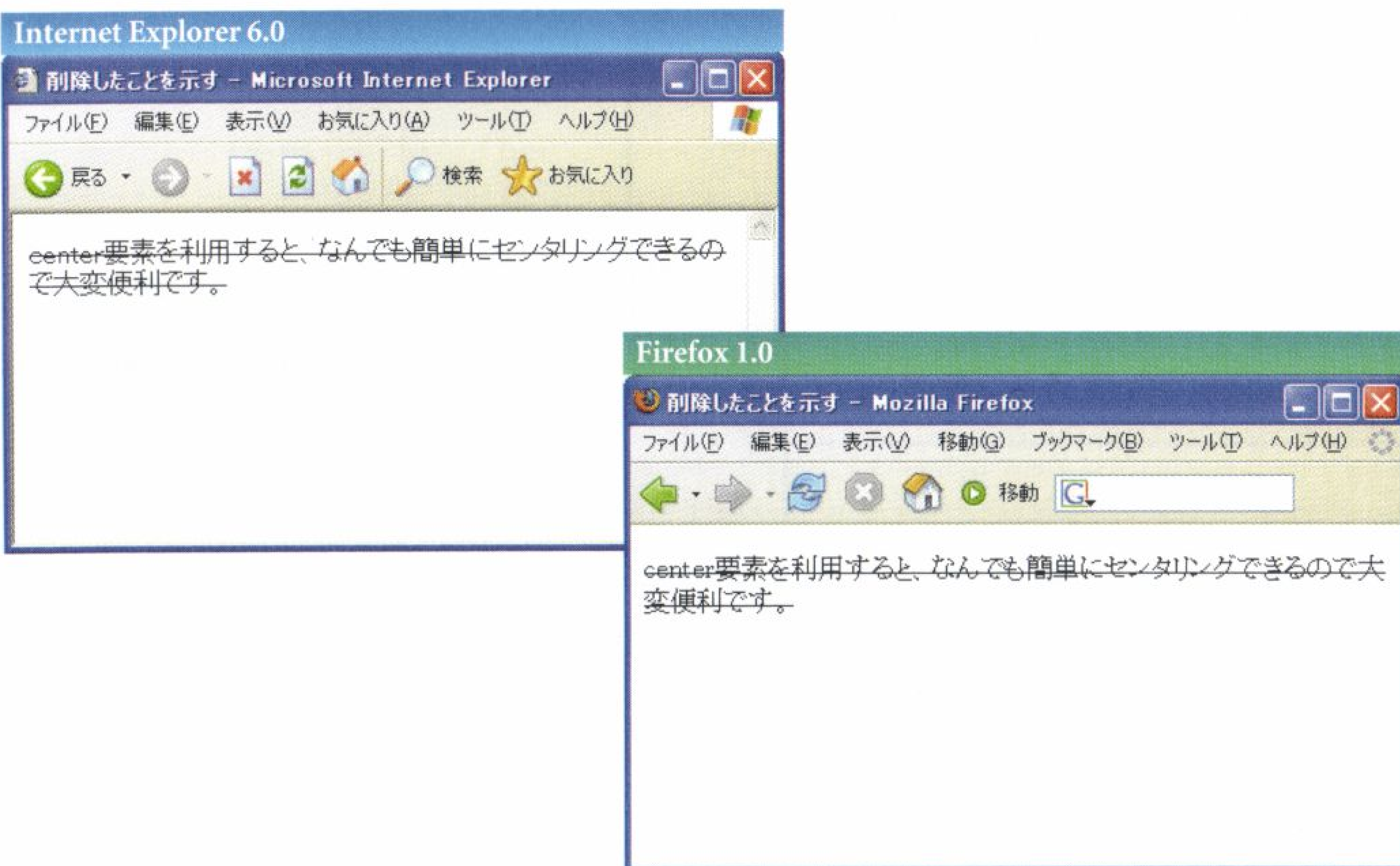
テキストの種類を指定する

# 削除したことを示す

**<del cite="*URL*" datetime="*削除日時*">～</del>**

| | |
|---|---|
| cite | 削除に関する情報が書かれている文書のURL |
| datetime | 削除した日時(ISO8601形式) |






del要素は、その部分が、後から削除された部分であることを示します。
ブラウザによって表示方法は異なりますが、一般的には取消線が付けられた状態で表示されます。また、ブラウザによっては、title属性で指定した短い説明をツールチップとして表示させることもできます。この要素は、指定する範囲に応じて、インライン要素としてもブロックレベル要素としても使用することができます。

## Sample

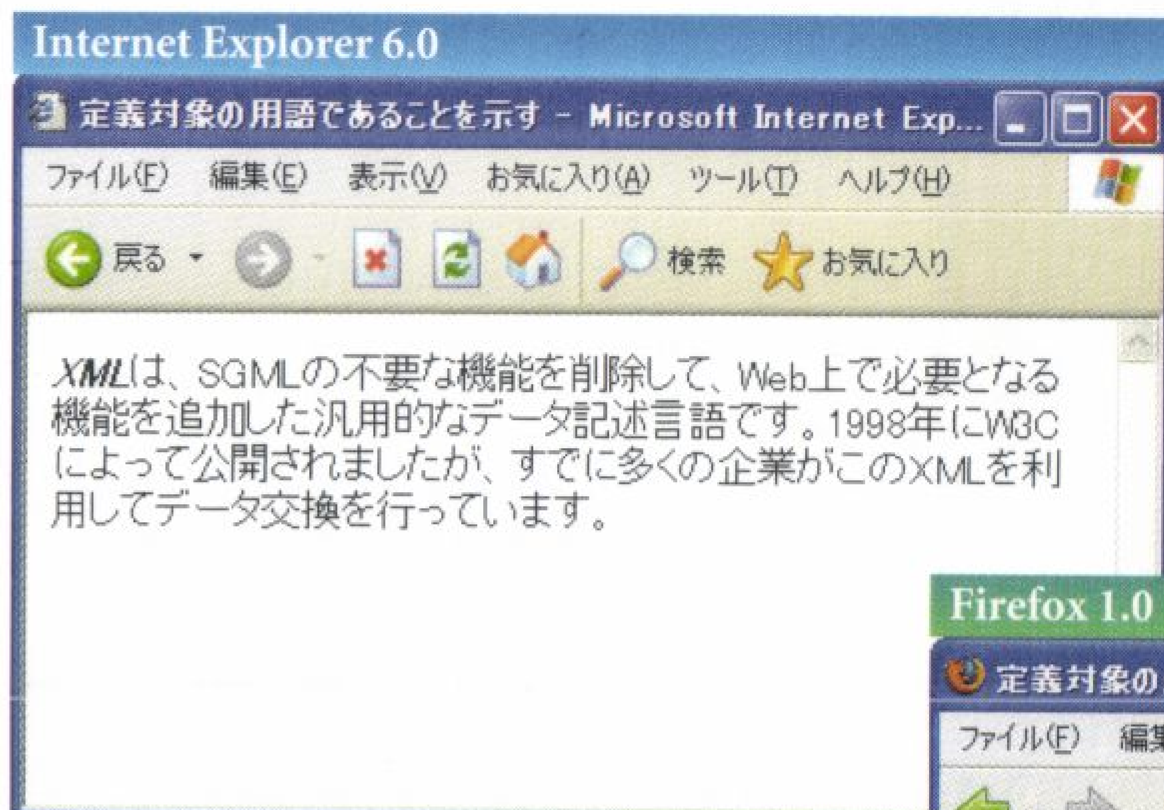
```
<p>
<del cite="http://www.w3.org/TR/xhtml11" datetime="2001-05-
31T21:30:00+09:00" title="スタイルシートを使用しましょう">
center要素を利用すると、なんでも簡単にセンタリングできるので大変便利です。
</del>
</p>
```


TIPS「追加や削除をした日時の表し方」(P.35)

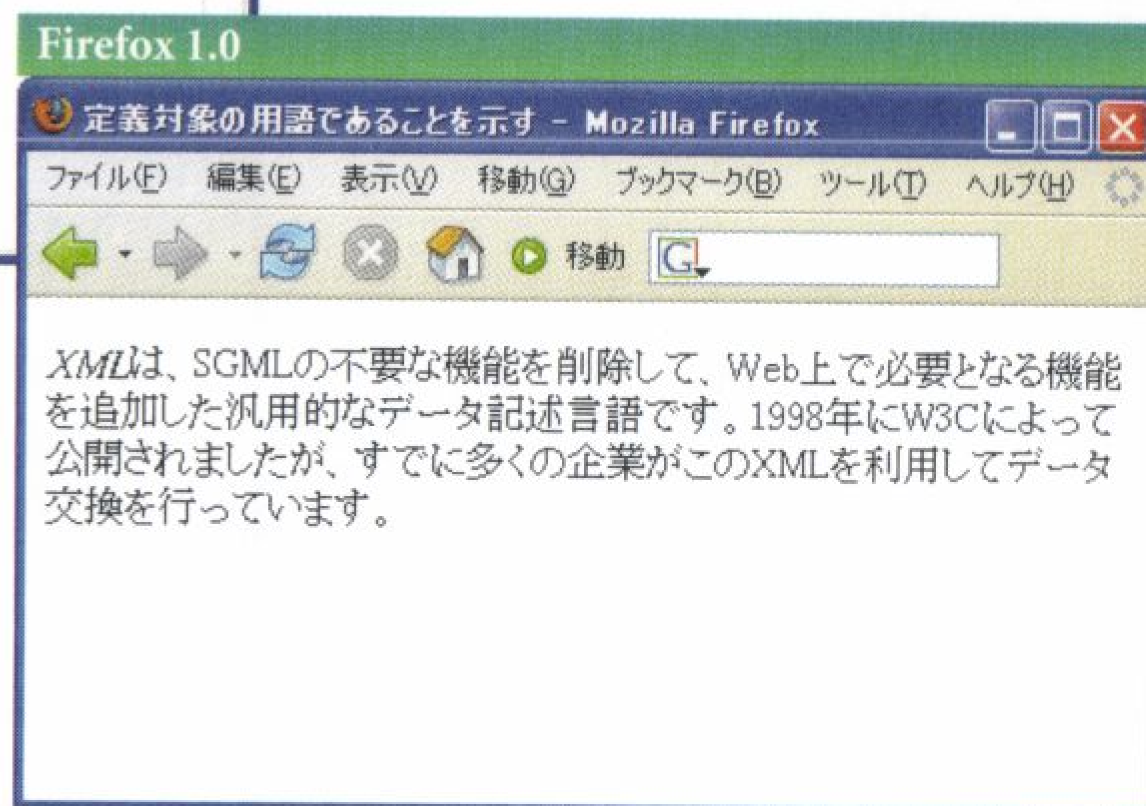
# 定義対象の用語であることを示す

## <dfn>～</dfn>

Internet Explorer 6.0




Firefox 1.0




dfn要素は、その部分が、用語の意味を定義(説明)している部分の対象となる「用語」あることを示します。
たとえば、ある文章の中で、初めてその用語が出てくる場合などに使用されます。一般的なブラウザでは、イタリックで表示されます。

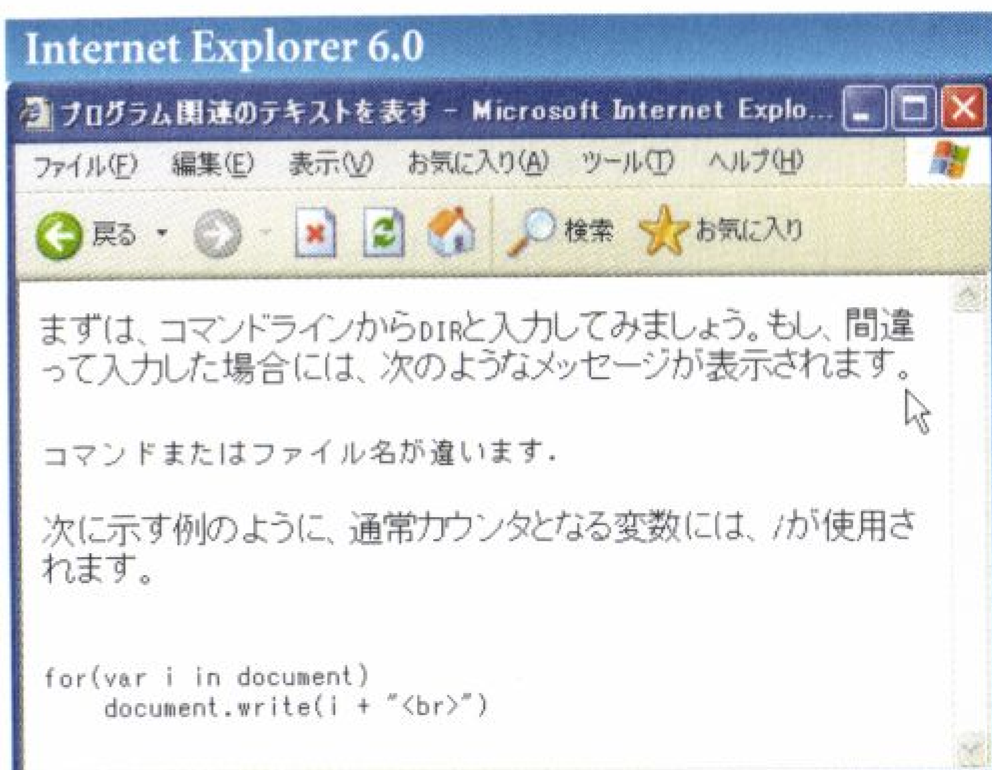
### Sample

```
<p>
<dfn>XML</dfn>は、SGMLの不要な機能を削除して、Web上で必要となる機能を追加した汎用的なデータ記述言語です。1998年にW3Cによって公開されましたが、すでに多くの企業がこのXMLを利用してデータ交換を行っています。
</p>
```

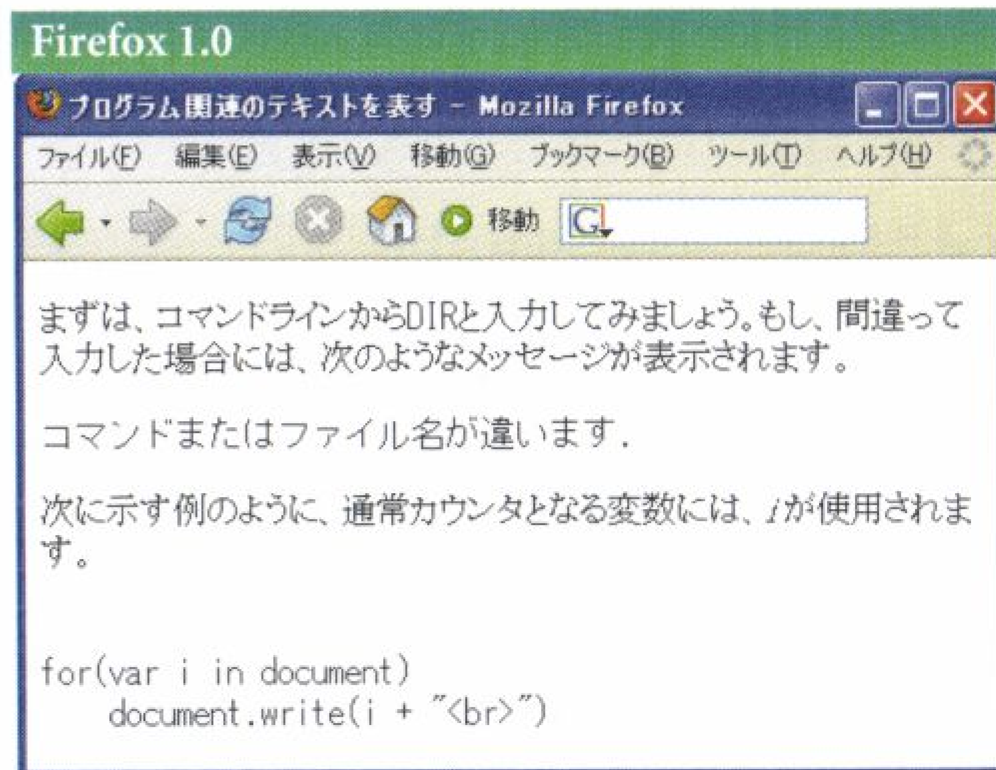
# プログラム関連のテキストを表す

**<kbd>～</kbd>** ←入力文字
**<samp>～</samp>** ←出力サンプル
**<code>～</code>** ←ソースコード
**<var>～</var>** ←変数・引数

Internet Explorer 6.0




Firefox 1.0




kbd要素は、その部分がキーボードなどから入力する文字であることを示します。samp要素は、プログラムなどによって出力される内容のサンプルを示す場合に使用します。code要素は、プログラムなどのソースコードを示す場合に使用します。ソースコード中の空白による字下げもそのまま表示させたい場合には、同時にpre要素も使用してください。var要素は、変数や引数を示す場合に使用します。
一般的なブラウザでは、kbd要素・samp要素・code要素は等幅フォントで、var要素はイタリックで表示されます。

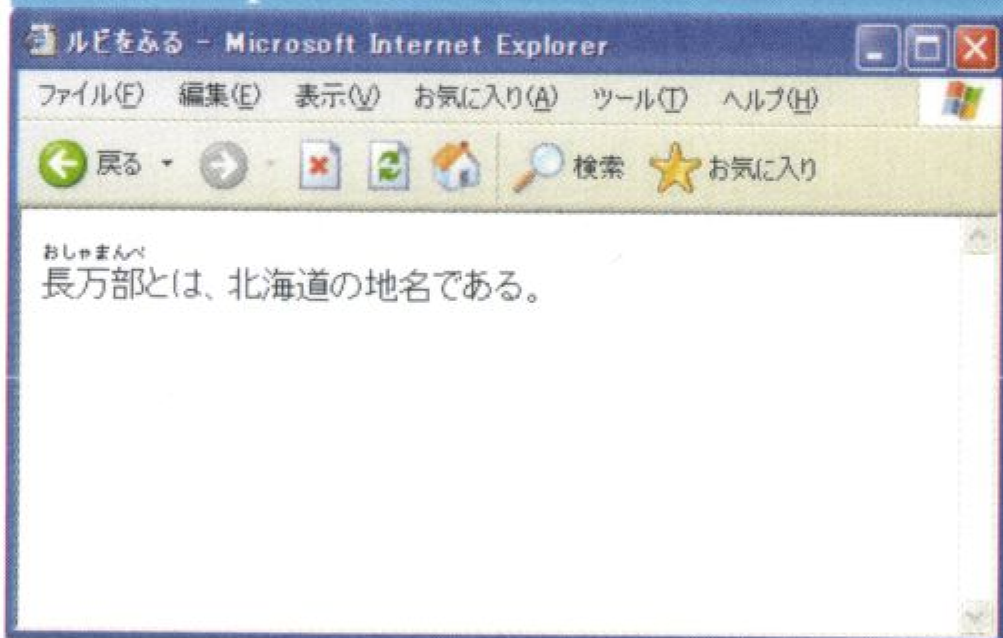
## Sample

```
<p>
まずは、コマンドラインから<kbd>DIR</kbd>と入力して
みましょう。もし、間違って入力した場合には、次の
ようなメッセージが表示されます。
</p>
<p>
<samp>コマンドまたはファイル名が違います.</samp>
</p>
  〜
<p>
次に示す例のように、通常カウンタとなる変数には、
<var>i</var> が使用されます。
</p>
<pre><code>
for(var i in document)
    document.write(i + "&lt;br&gt;")
</code></pre>
```

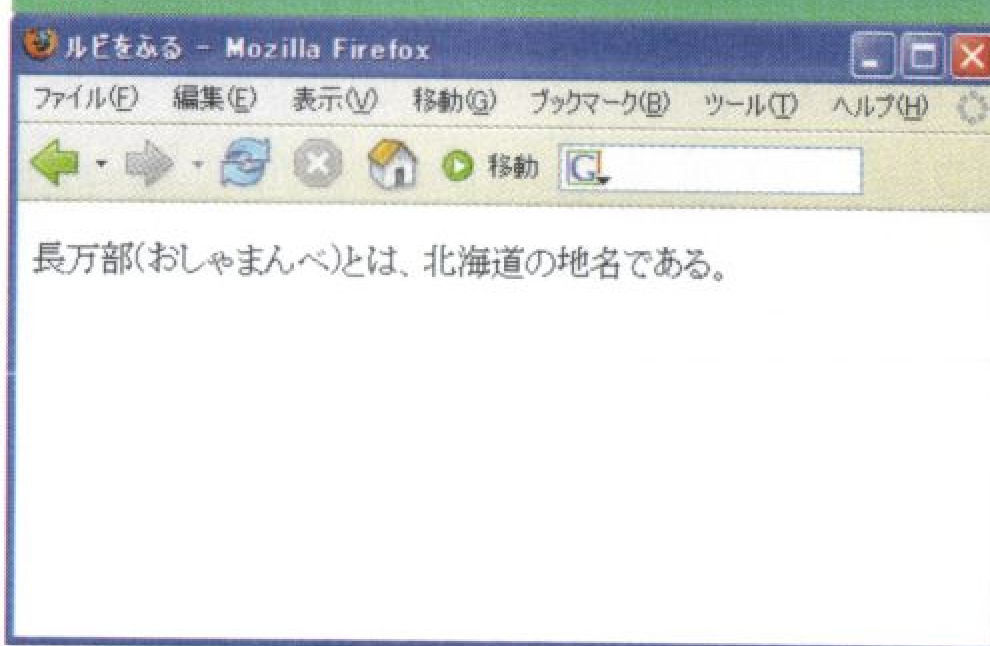
# ルビをふる

<ruby>～</ruby> ←ルビ関連の範囲全体
<rb>～</rb> ←ルビをふる対象
<rt>～</rt> ←ルビ(ふりがな)
<rp>～</rp> ←ルビ未対応用のカッコ

Internet Explorer 6.0




Firefox 1.0




指定した範囲にルビ(ふりがな)をふります。ルビに関連する要素は、すべて<ruby>～</ruby>の中に配置します。ルビをふる対象の言葉はrb要素で、ルビ自体はrt要素で示し、ルビに対応していないブラウザに備えて、ルビとなる文字(rt要素)の前後に内容としてカッコを入れたrp要素を配置します。rp要素の内容は、ルビに対応したブラウザでは無視されます。

なお、ルビはInternet Explorerのバージョン5以降で利用できますが、仕様としてはXHTML1.1から利用できるようになったものです。HTML4.01やXHTML1.0の文書では、文法的にはルビを使用できないということも覚えておいてください。

## Sample

```
<p>
<ruby>
<rb>長万部</rb>
<rp>(</rp>
<rt>おしゃまんべ</rt>
<rp>)</rp>
</ruby>
とは、北海道の地名である。
</p>
```

# 特別な文字を表示させる

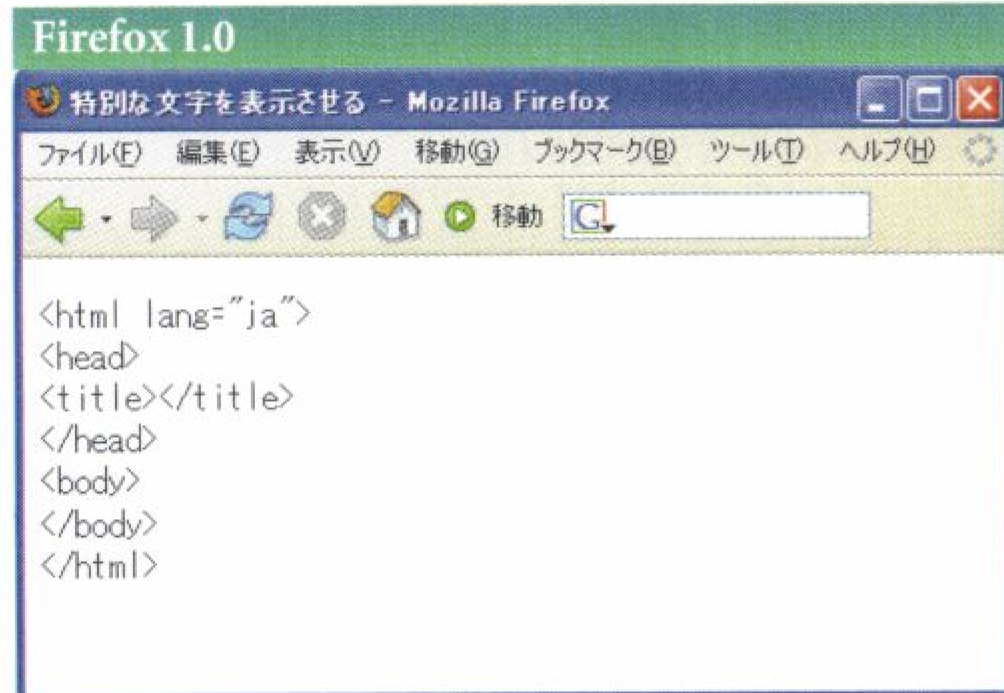
**<** ← <
**>** ← >
**"** ← "
**&** ← &



Internet Explorer 6.0




Firefox 1.0




HTMLでは、「<」と「>」はタグを表すために使用されますので、そのまま書くとそこがタグの一部だと解釈されてしまいます。そのような特別な意味を持った文字を表示させたい場合には、「&○○;」という形式を使用します。これらは、上の書式の通りに必ず小文字で書くようにしてください。実際には、この他にもさまざまな文字を表示させることが可能です。それらについては、P.174のリファレンス「HTML4.01で定義されている特別な文字」を参照してください。

## Sample

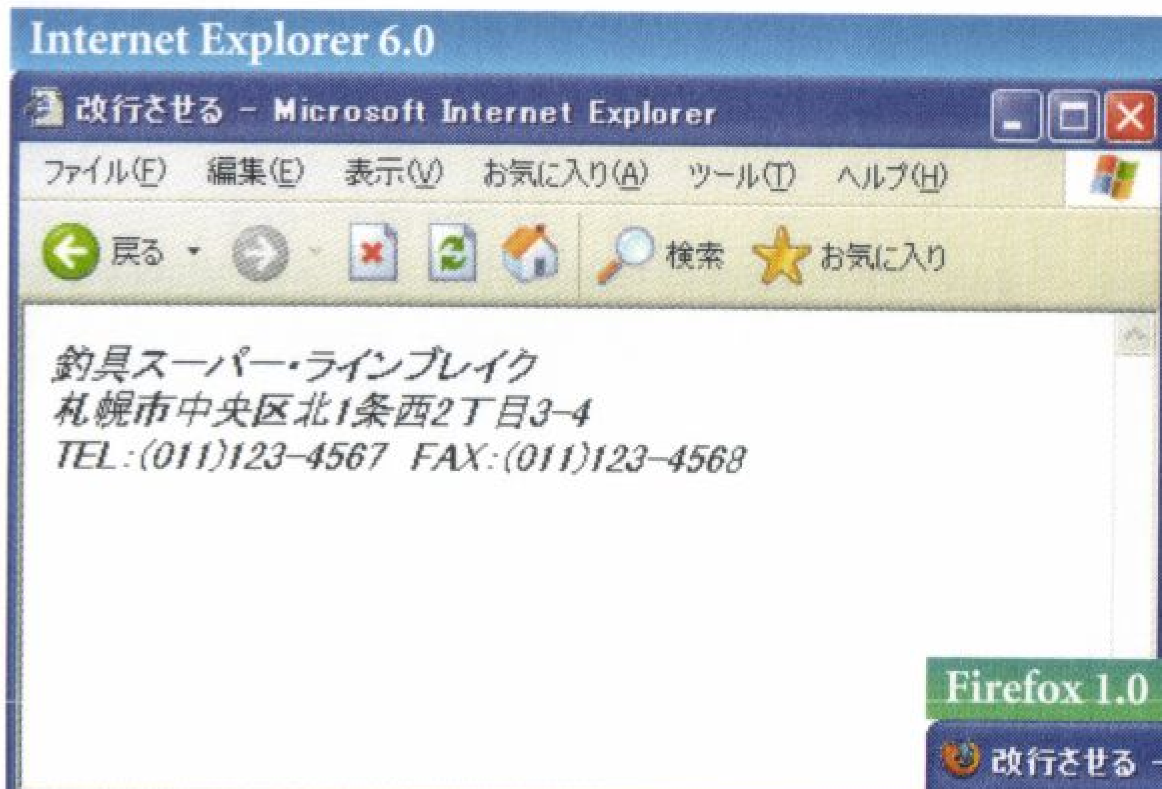
```
<p>
<code>
&lt;html lang=&quot;ja&quot;&gt;<br>
&lt;head&gt;<br>
&lt;title&gt;&lt;/title&gt;<br>
&lt;/head&gt;<br>
&lt;body&gt;<br>
&lt;/body&gt;<br>
&lt;/html&gt;
</code>
</p>
```

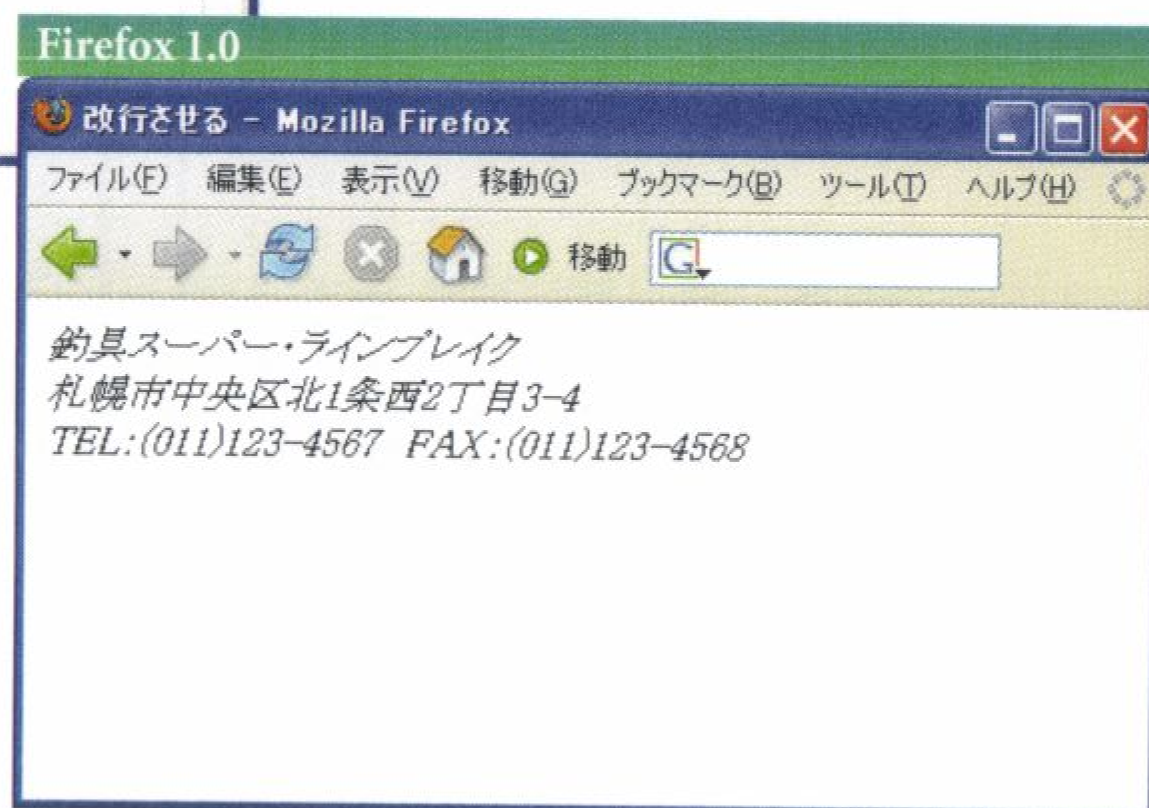

特別な文字：リファレンス「HTML4.01で定義されている特別な文字」（P.174）

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac i-mode

# 改行させる

## <br>




br要素を入れると、テキストがそこで改行されます。
HTMLのソースの中で改行してもブラウザで表示した場合には反映されませんので、ブラウザ上で改行させたい場合には、この要素を使用します。ただし、改行ができるからといって、この要素を使用して段落を表現するようなことは避けてください(段落を示すには、p要素を使ってください)。また、ユーザーは必ずしも制作者と同様のウィンドウ幅や文字サイズで見ているとは限りませんので、整形やレイアウトを目的として改行するような場合には、注意してください。

### Sample

```
<address>
釣具スーパー・ラインブレイク<br>
札幌市中央区北1条西2丁目3-4<br>
TEL：(011)123-4567　FAX：(011)123-4568
</address>
```

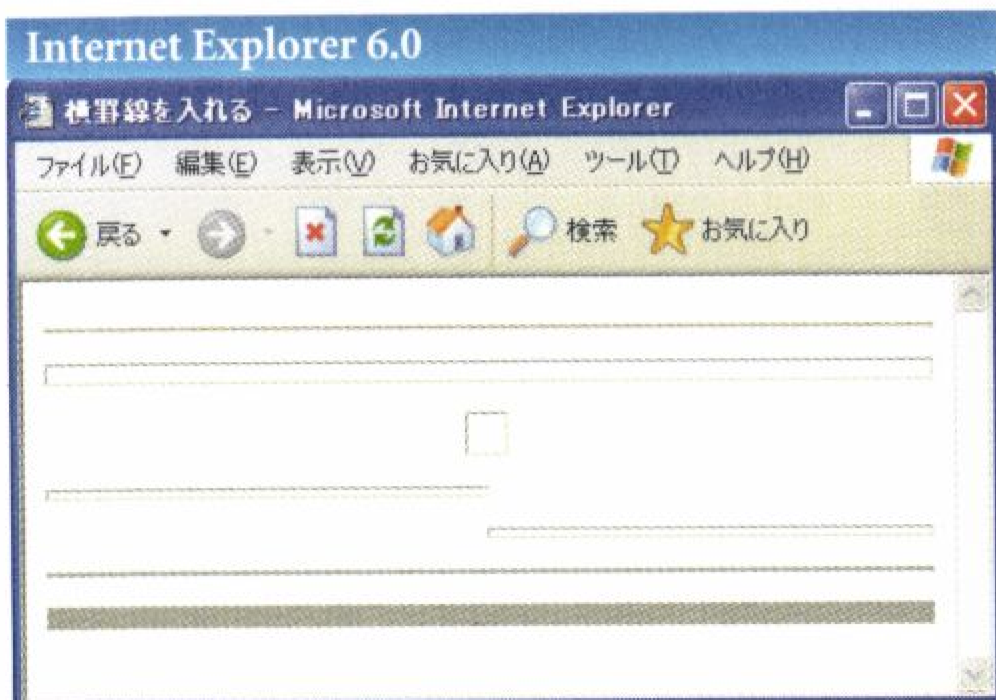
# 横罫線を入れる

**<hr>**
**<hr size="*太さ*" width="*長さ*" align="*行揃え位置*" *noshade*>**

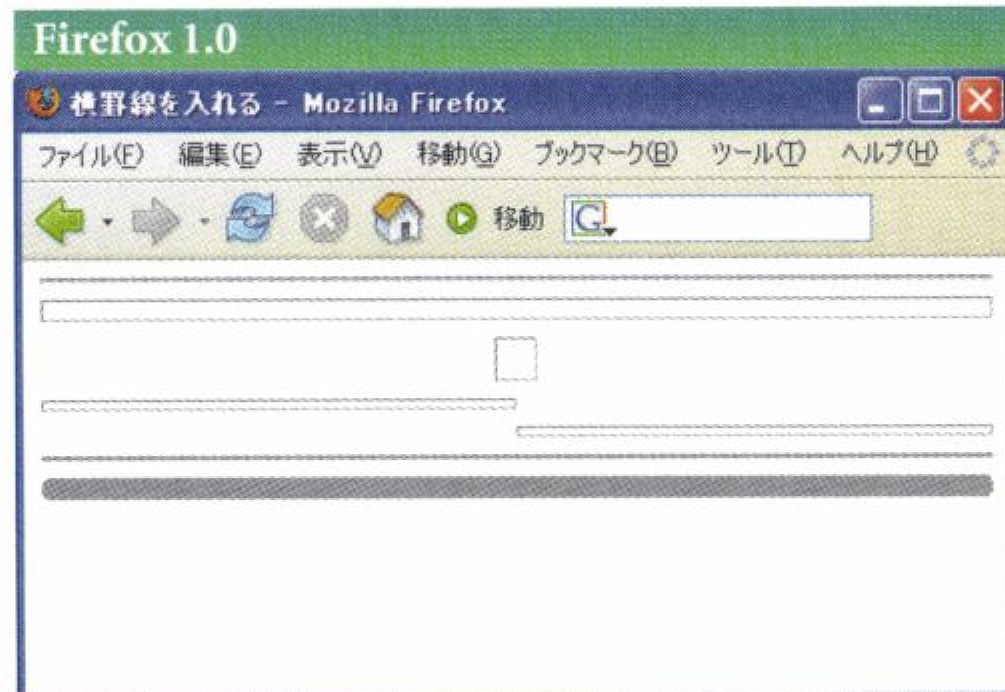
| | |
|---|---|
| size | 罫線の太さ（ピクセル） |
| width | 罫線の長さ（ピクセル・%） |
| align | 行揃え位置（left ・ right ・ center） |
| noshade | 罫線を平面的に表示 |



Internet Explorer 6.0




Firefox 1.0




hr要素を入れると、そこに横罫線が引かれます。
一般的なブラウザでは、引っ込んだ感じで立体的に表示されますが、noshade属性を指定すると単純な塗りつぶしの線にすることができます。size属性は線の太さを、width属性は線の長さを設定します。width属性は、単位を付けなければピクセル数として認識され、単位として「%」を付けるとウィンドウの幅に対する割合となります。align属性は罫線の行揃えを設定しますが、何も指定しない場合は一般に「center」になります。
ただし、これらの属性はすべて非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。表示方法を指定する場合には、できるだけスタイルシートを利用するようにしてください。スタイルシートでボックスの枠線を表示させると、hr要素を使用しなくても罫線を表示させることができます。



## Sample

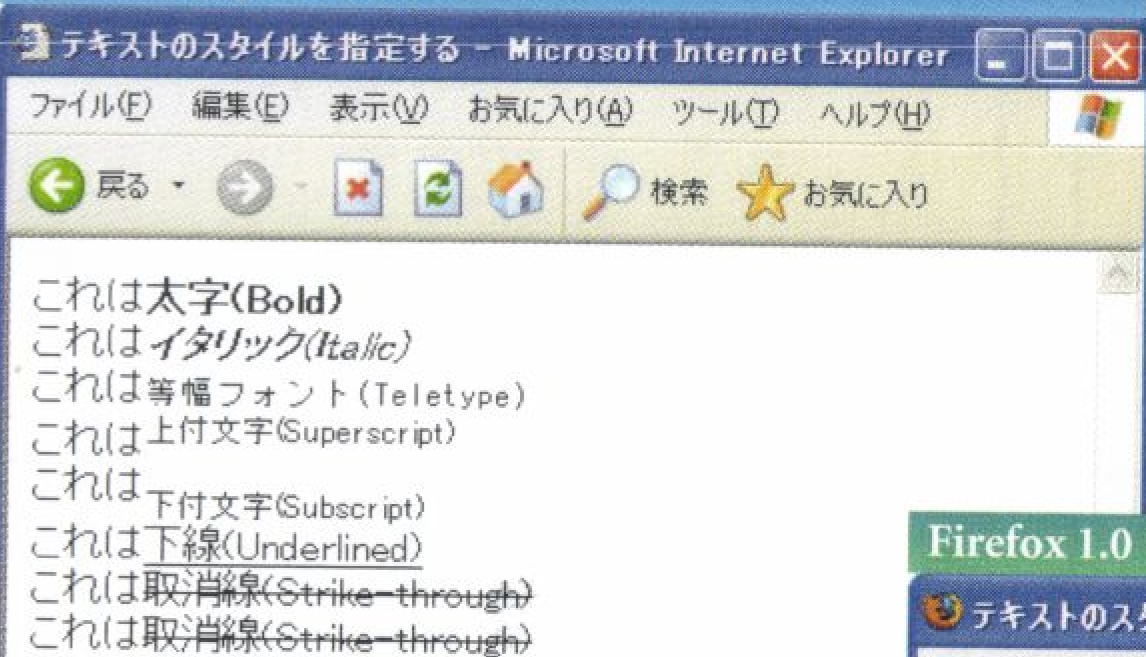
```
<hr>
<hr size="10">
<hr size="20" width="20">
<hr size="5"  width="50%" align="left">
<hr size="5"  width="50%" align="right">
<hr noshade>
<hr noshade size="10">
```


スタイルシート：「ボックス」の枠線に関連するプロパティ（P.241～248）

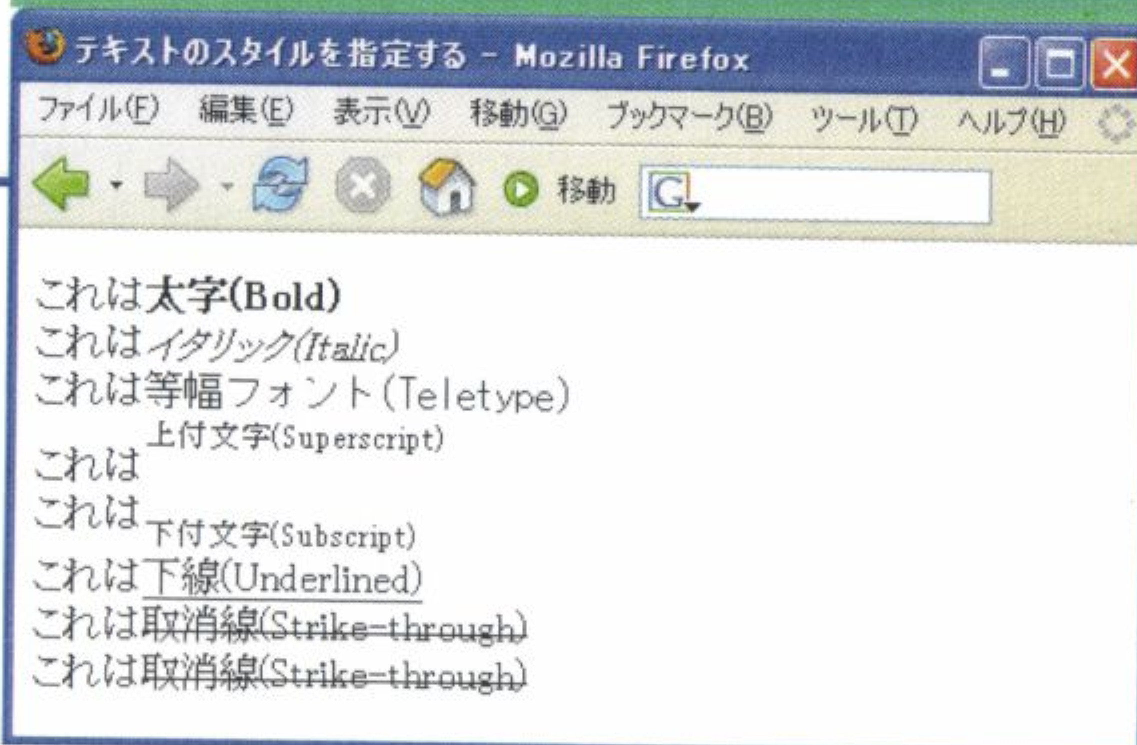
# テキストのスタイルを指定する

| | |
|---|---|
| <b>～</b> | ←太字 |
| <i>～</i> | ←イタリック |
| <tt>～</tt> | ←等幅フォント |
| <sup>～</sup> | ←上付文字 |
| <sub>～</sub> | ←下付文字 |
| <u>～</u> | ←下線 |
| <s>～</s> | ←取消線 |
| <strike>～</strike> | ←取消線 |

Internet Explorer 6.0




Firefox 1.0




指定した範囲のテキストのスタイルを指定します。
これらの要素は、文書構造としての意味は持ってはいませんが、HTMLでは該当する適切な要素がない場合や、ブラウザが使いたい要素をサポートしていない場合などに、補助的に使用すると便利です。
なお、これらの要素のうち、u要素・s要素・strike要素は非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。たとえば、強調のために下線を引きたいのであればem要素やstrong要素を、内容を取り消した(削除した)ことを示すために取消線を引きたいのであればdel要素を使用するなどして、どのように表示するかはスタイルシートで指定するようにしてください。

**Sample**

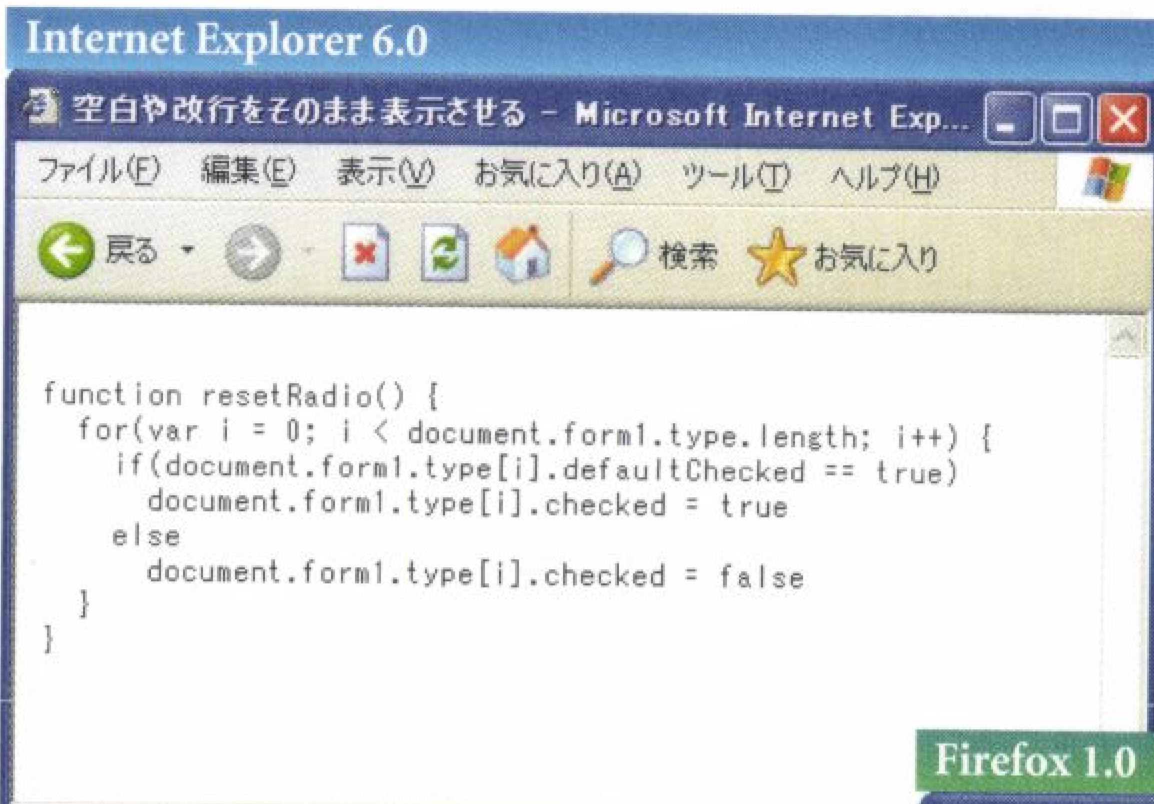
```
<p>
これは<b>太字(Bold)</b><br>
これは<i>イタリック(Italic)</i><br>
これは<tt>等幅フォント(Teletype)</tt><br>
これは<sup>上付文字(Superscript)</sup><br>
これは<sub>下付文字(Subscript)</sub><br>
これは<u>下線(Underlined)</u><br>
これは<s>取消線(Strike-through)</s><br>
これは<strike>取消線(Strike-through)</strike>
</p>
```

スタイルシート:「フォント」の「フォントを指定する」(P.202)
スタイルシート:「フォント」の「フォントの太さを指定する」(P.206)
スタイルシート:「フォント」の「フォントスタイルを指定する」(P.208)
スタイルシート:「テキスト」の「縦方向の位置を指定する」(P.215)

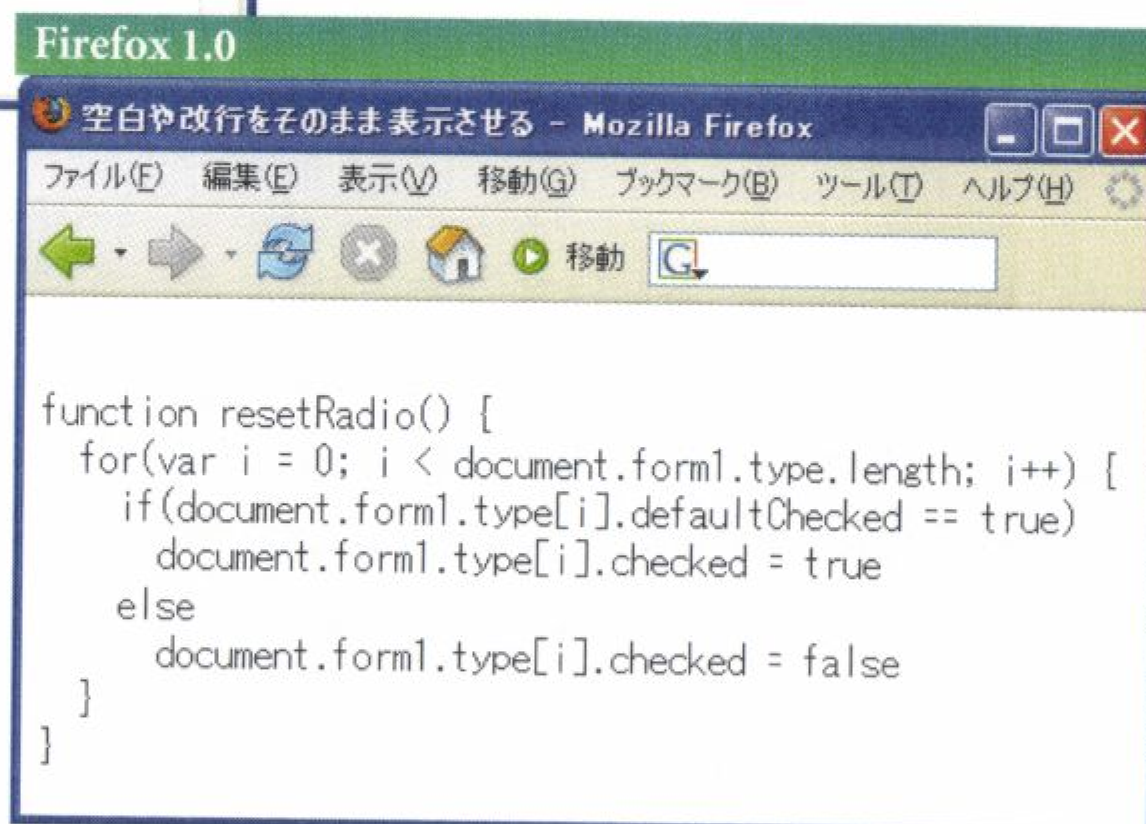
# 空白や改行をそのまま表示させる

## <pre>～</pre>

Internet Explorer 6.0




Firefox 1.0




pre要素は、指定した範囲のテキストを、入力した通りに等幅フォントで表示します。具体的には、スペースと改行が入力したままの状態で表示され、1行の長さがウィンドウの幅より長くなっても自動的に改行されなくなります。主にソースコードなどを表示させたい場合に使用しますが、タブは入れないようにしてください。

### Sample

```
<pre><code>
function resetRadio() {
  for(var i = 0; i &lt; document.form1.type.length; i++) {
    if(document.form1.type[i].defaultChecked == true)
      document.form1.type[i].checked = true
    else
      document.form1.type[i].checked = false
  }
}
</code></pre>
```

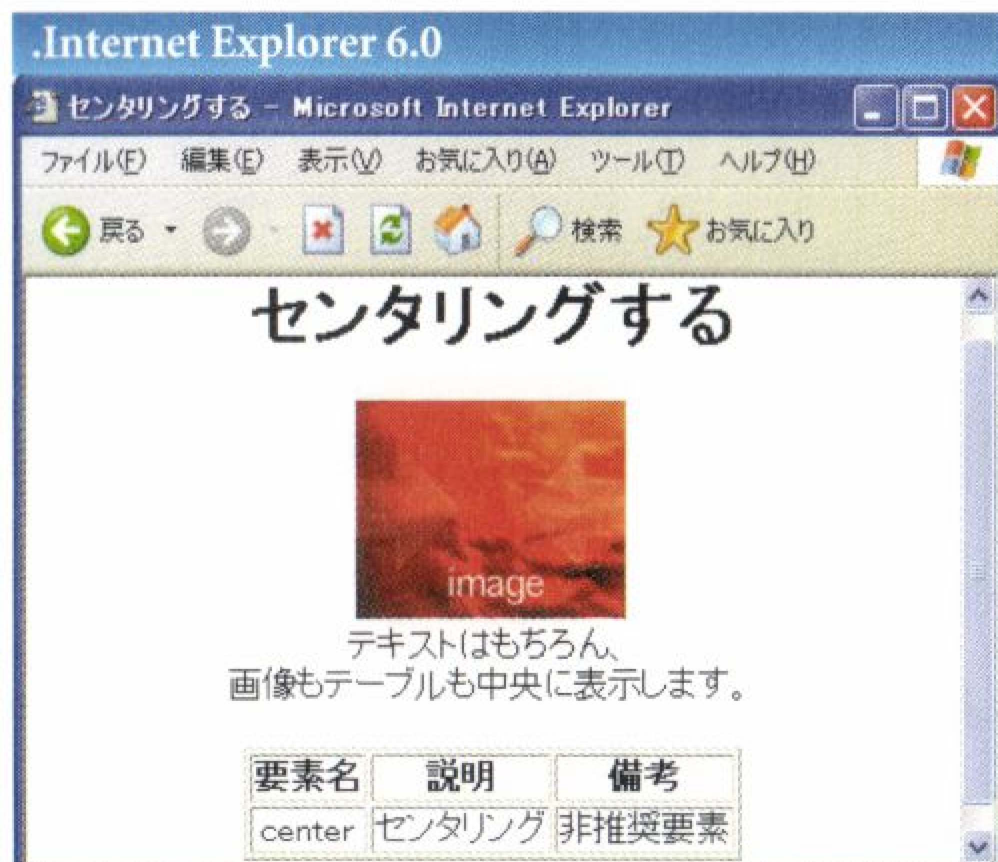

<code>：「テキストの種類」の「プログラム関連のテキストを表す」(P.39)

# センタリングする

## <center>～</center>



.Internet Explorer 6.0




Firefox 1.0




center 要素は、指定した範囲の内容をセンタリングして表示します。
ブロックレベル要素かインライン要素かを問わずに、ほとんどの要素をセンタリングすることができます。
ただし、この要素は非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。センタリングしたい場合には、できるだけスタイルシートを利用するようにしてください。

### Sample

```
<center>
<h1>センタリングする</h1>
<p>
<img src="image.jpg" width="128" height="100" alt=""><br>
テキストはもちろん、<br>画像もテーブルも中央に表示します。
</p>
<table border="1">
<tr>
<th>要素名</th><th>説明</th><th>備考</th>
</tr>
<tr>
<td>center</td><td>センタリング</td><td>非推奨要素</td>
</tr>
</table>
</center>
```


<table align="center">：「テーブル」の「表をセンタリングする」(P.94)
スタイルシート：「テキスト」の「行揃えを指定する」(P.214)
スタイルシート：「表示と配置」の「センタリングする」(P.256)

HTML

# 行揃えを指定する

**<h1 align="*行揃え位置*">～</h1>**
**<p align="*行揃え位置*">～</p>**
**<div align="*行揃え位置*">～</div>**

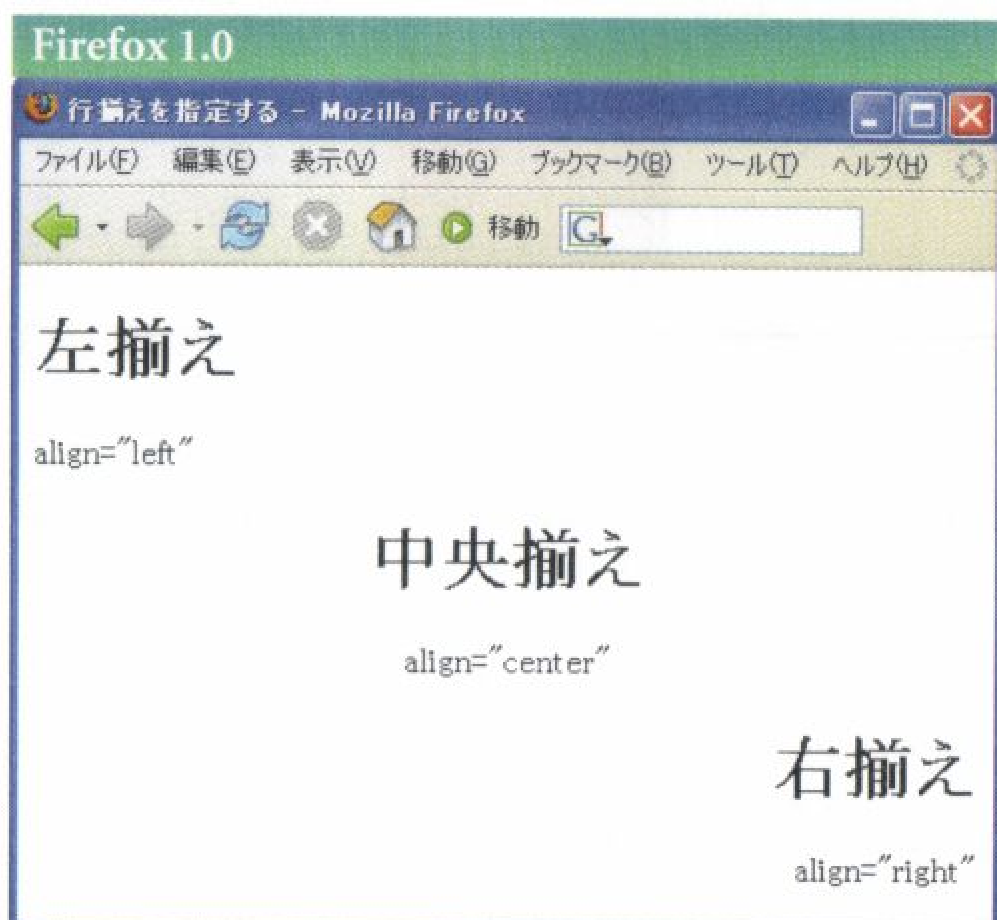
| 行揃え位置 | left · right · center |
|---|---|

※h1 だけでなく、h1～h6 で利用できます。

Internet Explorer 6.0




Firefox 1.0




align 属性を使用すると、見出し(h1～h6 要素)や段落(p 要素)、指定した範囲(div 要素)の行揃えを指定することができます。
行揃え位置として指定する「left」「right」「center」は、それぞれ「左揃え」「右揃え」「中央揃え」を表しています。
ただし、このalign属性を使用することは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。行揃えを指定する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<h1 align="left">左揃え</h1>
<p  align="left">align="left"</p>
<h1 align="center">中央揃え</h1>
<p  align="center">align="center"</p>
<h1 align="right">右揃え</h1>
<p  align="right">align="right"</p>
```


<div>: 「基本的な内容」の「目的に応じて範囲を指定する」(P.16)
<h1>: 「テキスト」の「見出しを表す」(P.27)
<p>: 「テキスト」の「段落を表す」(P.29)
スタイルシート: 「テキスト」の「行揃えを指定する」(P.214)

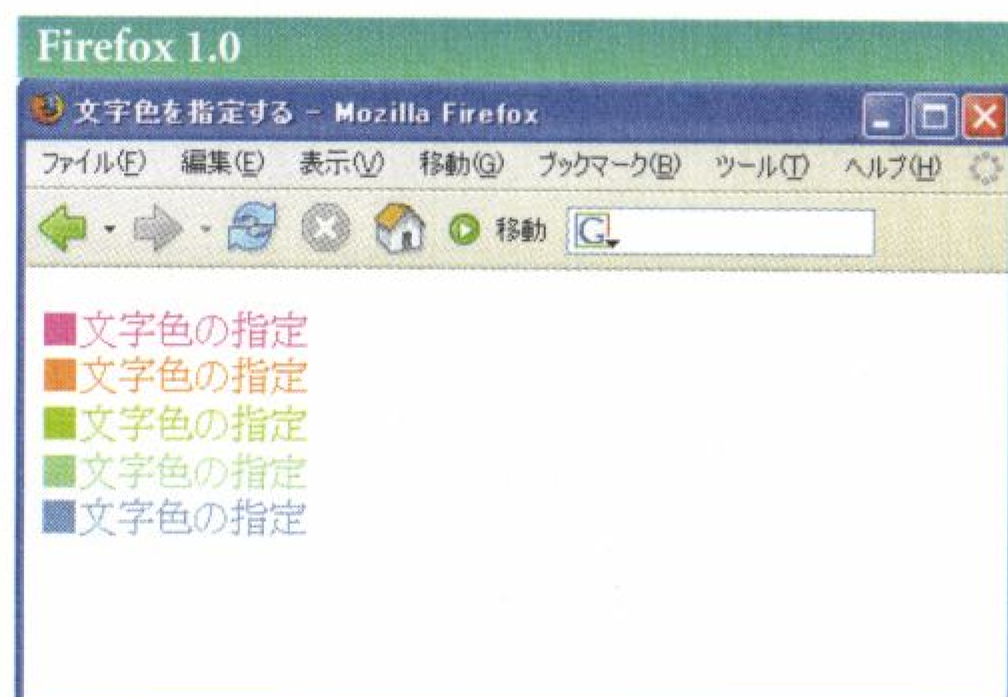
# 文字色を指定する

**<font color="色指定">～</font>**

Internet Explorer 6.0




Firefox 1.0




font要素のcolor属性は、指定した範囲のテキストの色を指定します。
ページ全体を通しての文字色やリンク部分に関する色は、body要素の属性で指定することができます。
ただし、HTMLで色の指定を行うことは非推奨とされており、新しいHTMLの標準仕様では指定できなくなっています。文字色を指定する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

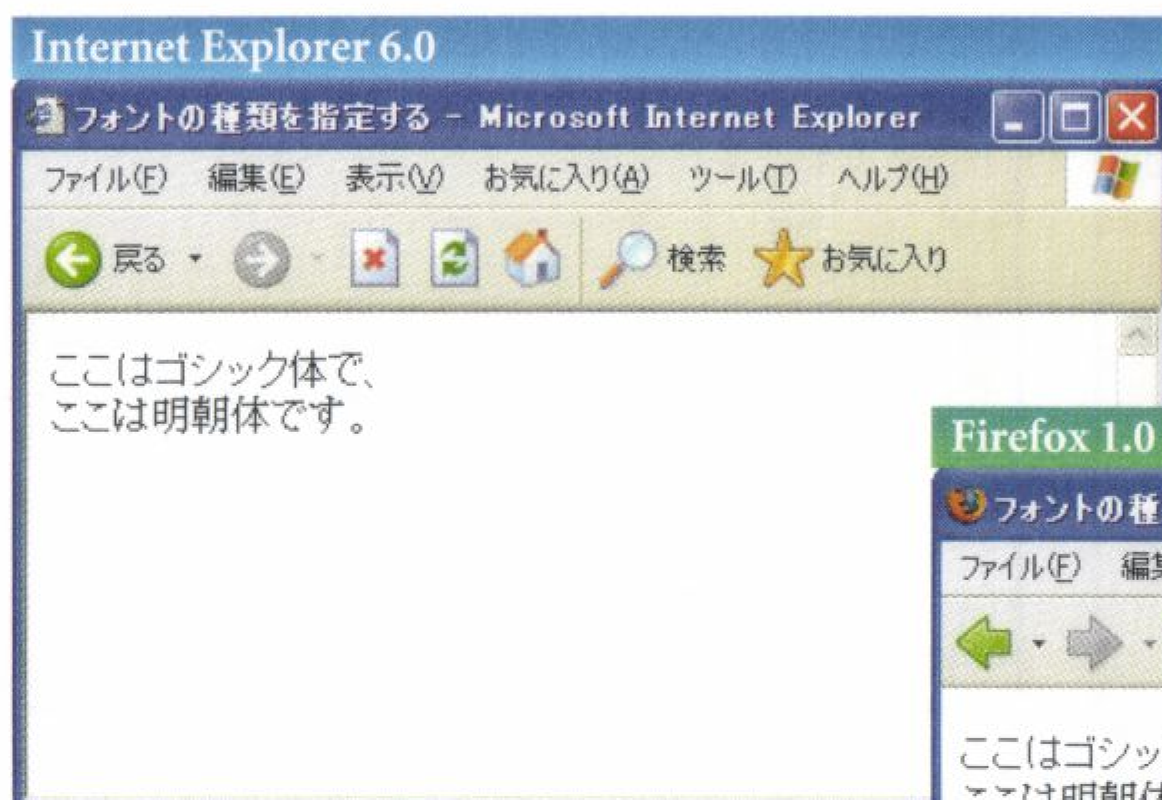
```
<p>
<font color="#ff0099">■文字色の指定</font><br>
<font color="#ff9900">■文字色の指定</font><br>
<font color="#99ff00">■文字色の指定</font><br>
<font color="#00ff99">■文字色の指定</font><br>
<font color="#0099ff">■文字色の指定</font>
</p>
```

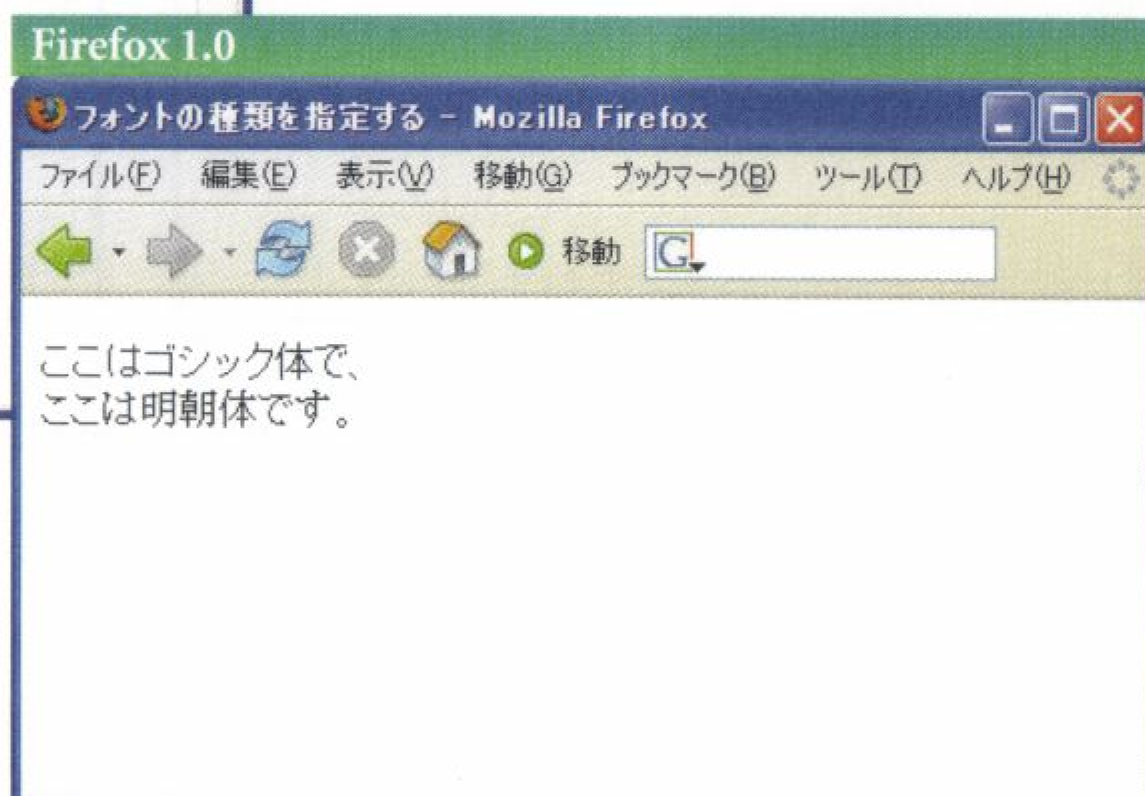
色指定：「HTMLについて」の「色の指定方法」(P.6)
<body text="～">：「基本的な内容」の「全体の文字色を設定する」(P.12)
<body link="～">：「リンク」の「リンク部分の文字色を設定する」(P.59)
スタイルシート：「フォント」の「文字色を指定する」(P.201)
色指定：巻末付録「カラーチャート1～3」(巻末)

# フォントの種類を指定する

## <font face="フォント名,フォント名,…">～</font>






font要素のface属性は、指定した範囲のテキストを表示するフォントの種類を指定します。

フォント名はひとつでも指定できますが、カンマで区切って複数指定しておくこともできます。その場合は、より先に指定されているフォントで、ユーザーの環境で表示可能なものが採用されます。フォント名が間違っていると正しく表示されませんので、注意してください(特に全角と半角の違いなど)。

ただし、font要素を使うことは非推奨とされており、新しいHTMLの標準仕様では使用できなくなっています。フォントの種類を指定する場合には、できるだけスタイルシートを利用するようにしてください。

### Sample

```
<p>
<font face="ＭＳ Ｐゴシック,中ゴシックＢＢＢ,Osaka">
ここはゴシック体で、</font>
<br>
<font face="ＭＳ Ｐ明朝,リュウミンライトーＫＬ,細明朝体">
ここは明朝体です。</font>
</p>
```

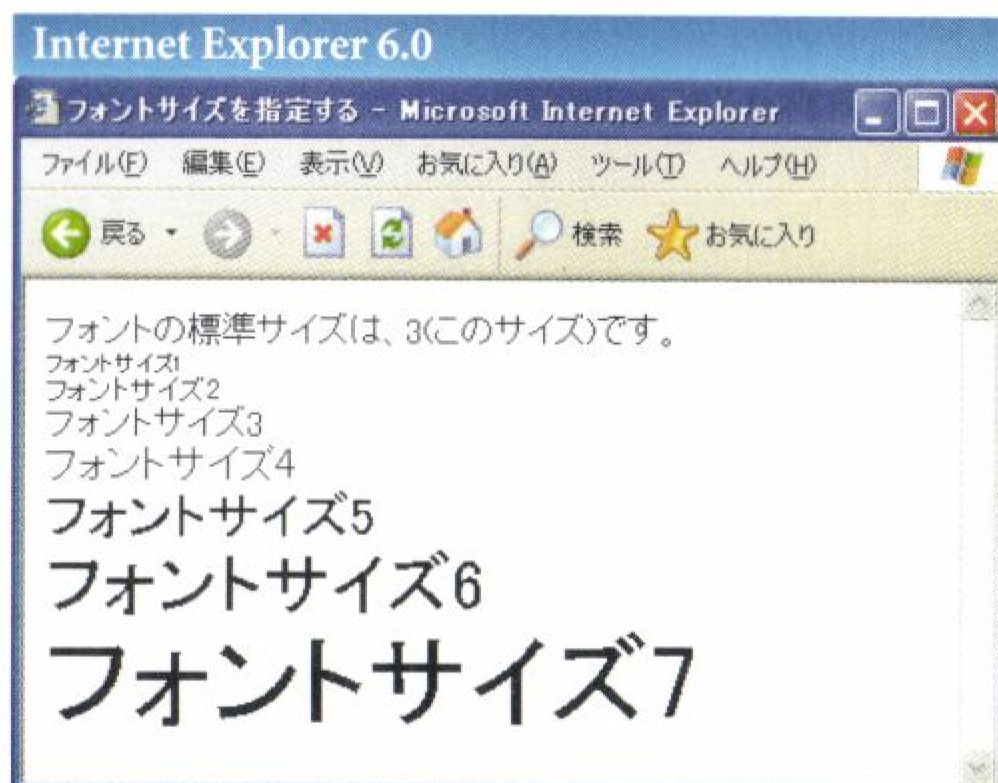
スタイルシート:「フォント」の「フォントを指定する」(P.202)
フォント名:巻末付録「フォント表示見本」(巻末)

# フォントサイズを指定する

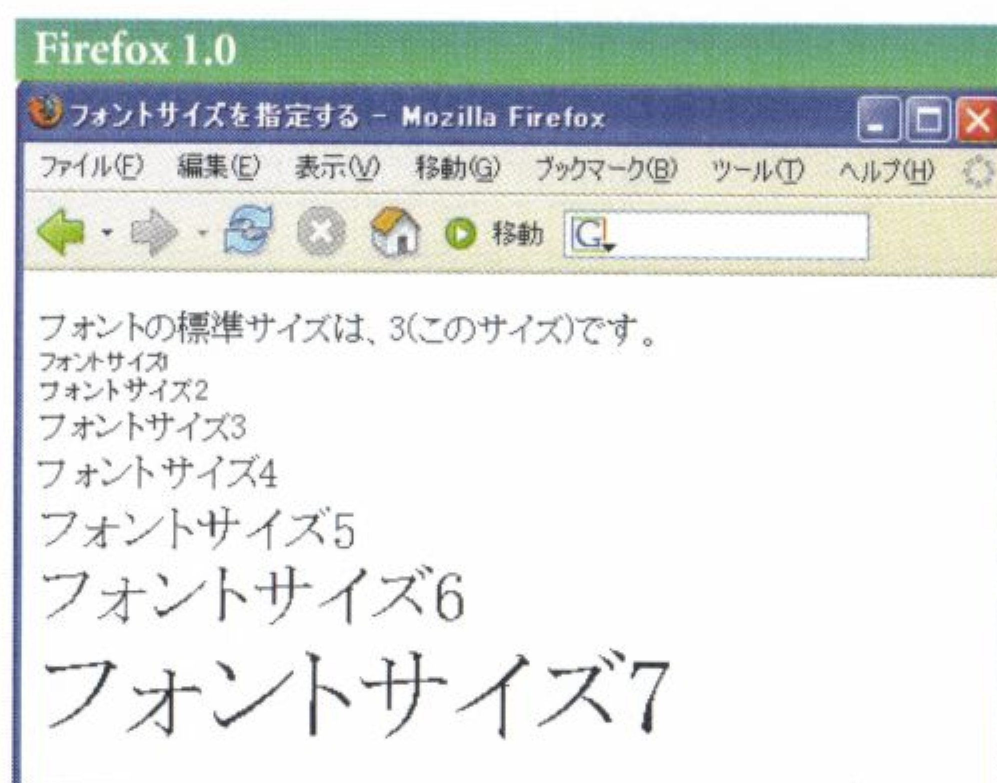
<font size="サイズ">～</font>

| size | 1～7（1が最小、7が最大。標準は3） |
|---|---|

Internet Explorer 6.0




Firefox 1.0




font要素のsize属性は、指定した範囲のフォントサイズを指定します。
サイズとしては1～7の数字を指定できますが、具体的な大きさが決まっているわけではないため、実際に表示される大きさはブラウザによって異なる場合があります。
ただし、font要素は非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。フォントサイズを指定する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<p>
フォントの標準サイズは、3（このサイズ）です。<br>
<font size="1">フォントサイズ1</font><br>
<font size="2">フォントサイズ2</font><br>
<font size="3">フォントサイズ3</font><br>
<font size="4">フォントサイズ4</font><br>
<font size="5">フォントサイズ5</font><br>
<font size="6">フォントサイズ6</font><br>
<font size="7">フォントサイズ7</font>
</p>
```


スタイルシート：「フォント」の「フォントサイズを指定する」（P.204）

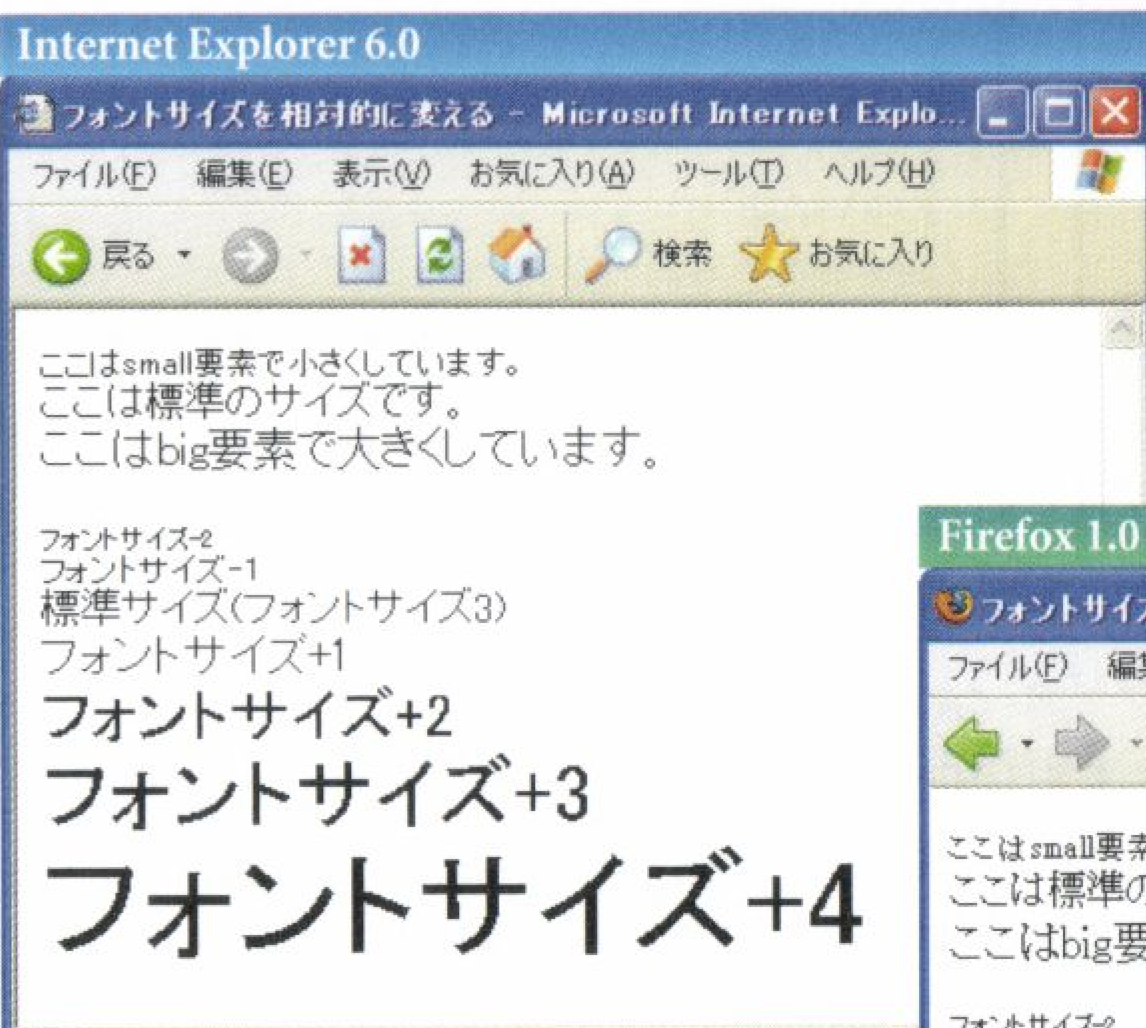
HTML

# フォントサイズを相対的に変える

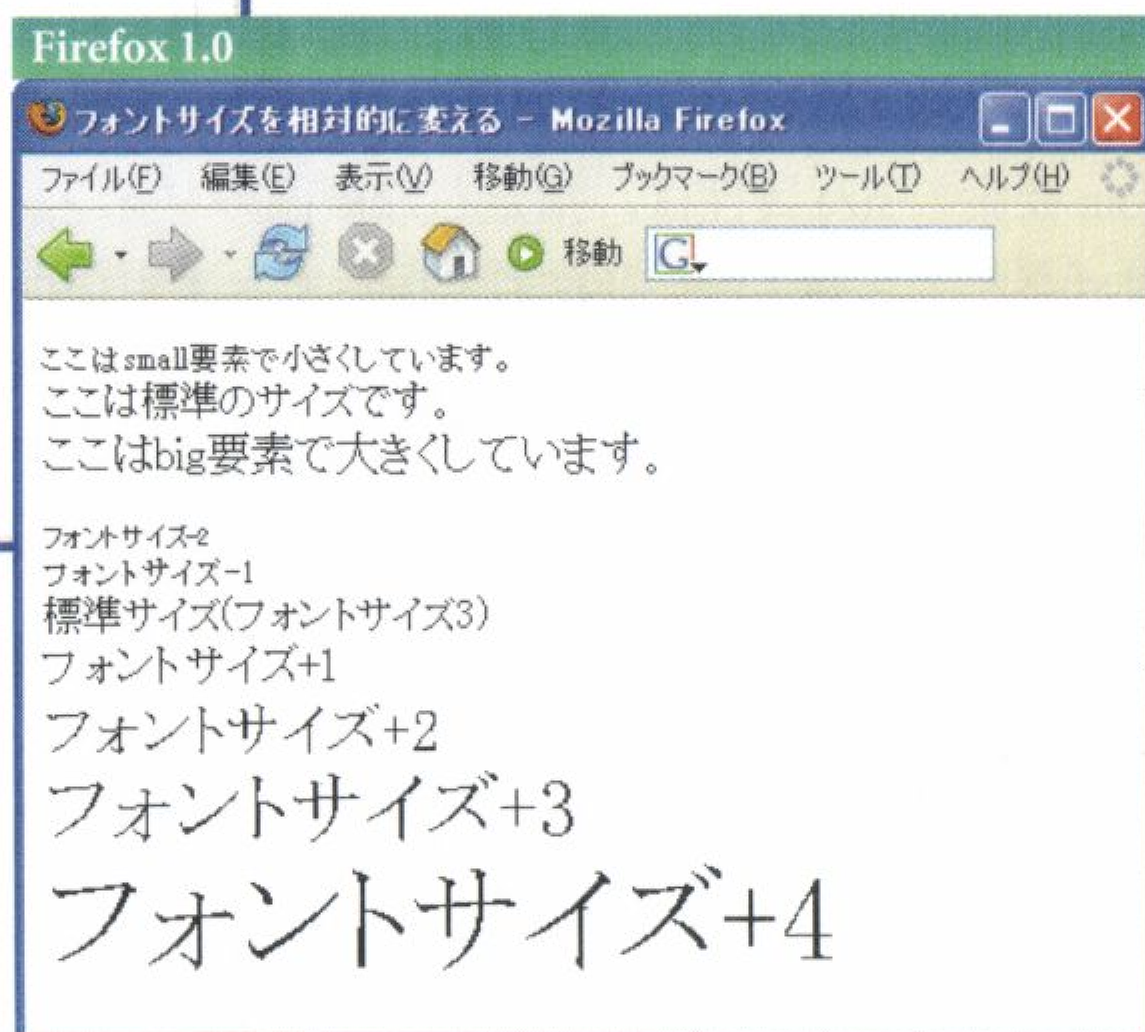
| | |
|---|---|
| <big>～</big> | ◀大きく |
| <small>～</small> | ◀小さく |
| <font size="+n">～</font> | ◀n段階大きく |
| <font size="-n">～</font> | ◀n段階小さく |

| | |
|---|---|
| n | 現在のサイズを基準として±の結果が1～7の範囲となる数字 |

Internet Explorer 6.0




Firefox 1.0




big要素はフォントサイズを大きく、small要素はフォントサイズを小さくして表示します。その際に、具体的にどの程度サイズを変更するのかは決まっていません。
font要素を使用すると、現在のサイズに対して何段階大きくまたは小さくするかを指定することができます。size属性に「+」または「-」を付けた数字で指定してください。ただし、フォントサイズは全部で7段階しかありませんので、±した結果が1～7の範囲になるようにしてください。
ただし、font要素は非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。フォントサイズを変更する場合には、できるだけスタイルシートを利用するようにしてください。

**Sample**

```
<p>
<small>ここはsmall要素で小さくしています。</small><br>
ここは標準のサイズです。<br>
<big>ここはbig要素で大きくしています。</big>
</p>
<p>
<font size="-2">フォントサイズ-2</font><br>
<font size="-1">フォントサイズ-1</font><br>
標準サイズ(フォントサイズ3)<br>
<font size="+1">フォントサイズ+1</font><br>
<font size="+2">フォントサイズ+2</font><br>
<font size="+3">フォントサイズ+3</font><br>
<font size="+4">フォントサイズ+4</font>
</p>
```

スタイルシート:「フォント」の「フォントサイズを指定する」(P.204)

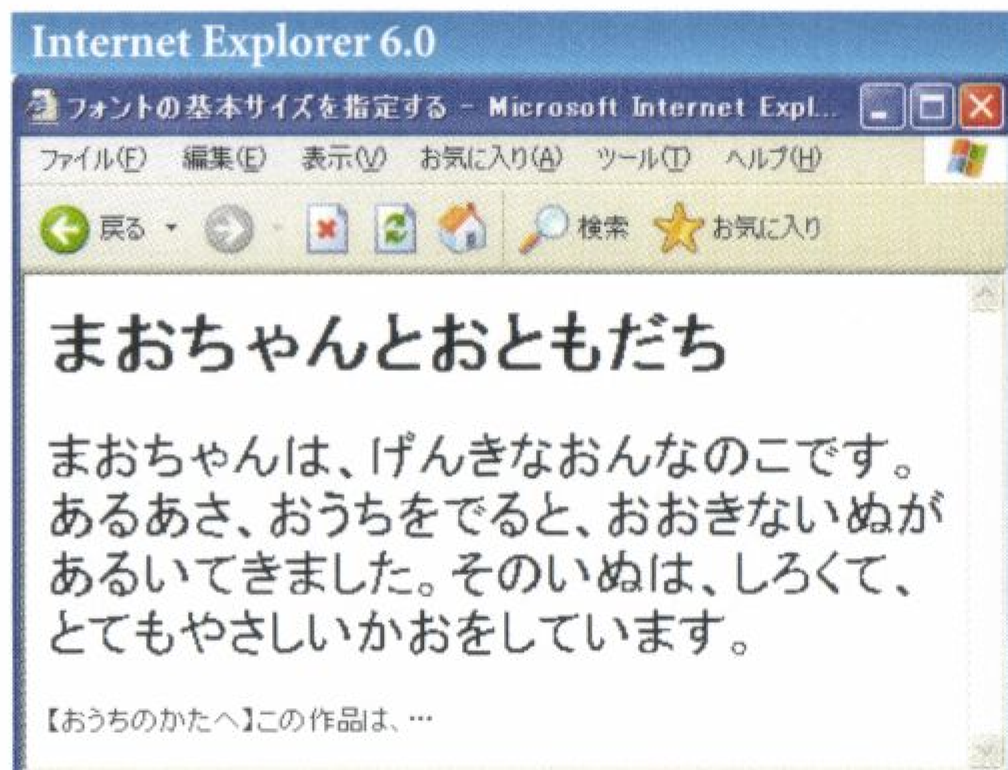
IE6.0
IE5.5
IE5.0
IE4.0
N4.X
IE5-mac
IE4-mac

# フォントの基本サイズを指定する

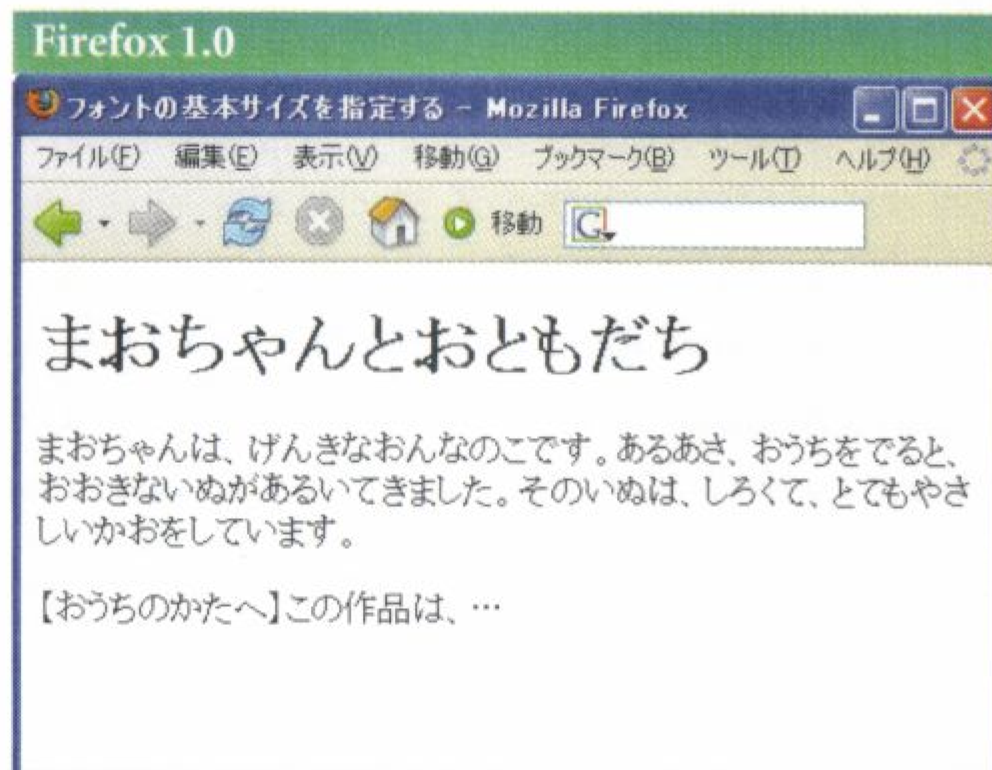
**<basefont size="サイズ">**

| | |
|---|---|
| size | 1～7(1が最小、7が最大。標準は3) |

Internet Explorer 6.0




Firefox 1.0




basefont要素は、この指定以降のフォントの基本サイズを設定します。
ただし、見出しの大きさには影響しません。サイズとしては1～7の数字を指定できますが、具体的な大きさが決まっているわけではないため、実際に表示される大きさはブラウザによって異なる場合があります。
ただし、basefont要素は非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。フォントのサイズに関する指定をする場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<basefont size="5">
<h1>まおちゃんとおともだち</h1>
<p>
まおちゃんは、げんきなおんなのこです。あるあさ、おうちをでると、おおきないぬがあるいてきま
した。そのいぬは、しろくて、とてもやさしいかおをしています。
</p>
  ⟅
<basefont size="2">
<p>
【おうちのかたへ】この作品は、…
</p>
```


スタイルシート：「フォント」の「フォントサイズを指定する」(P.204)

表示方法を指定する

# 他のページにリンクする

## <a href="リンク先URL">～</a>



Internet Explorer 6.0




Firefox 1.0




a要素のhref属性は、指定した範囲を他のページにリンクするようにします。
リンクさせる部分の言葉は、その部分だけを見てもリンク先の内容が具体的に連想できるようなものにしてください。「ここ」や「ここをクリック」というようなリンクの仕方は避けたほうがよいでしょう。また、画像をリンクさせる場合には、リンク先が明確にわかるような代替テキスト(alt="～")を必ず入れるようにしてください。

### Sample

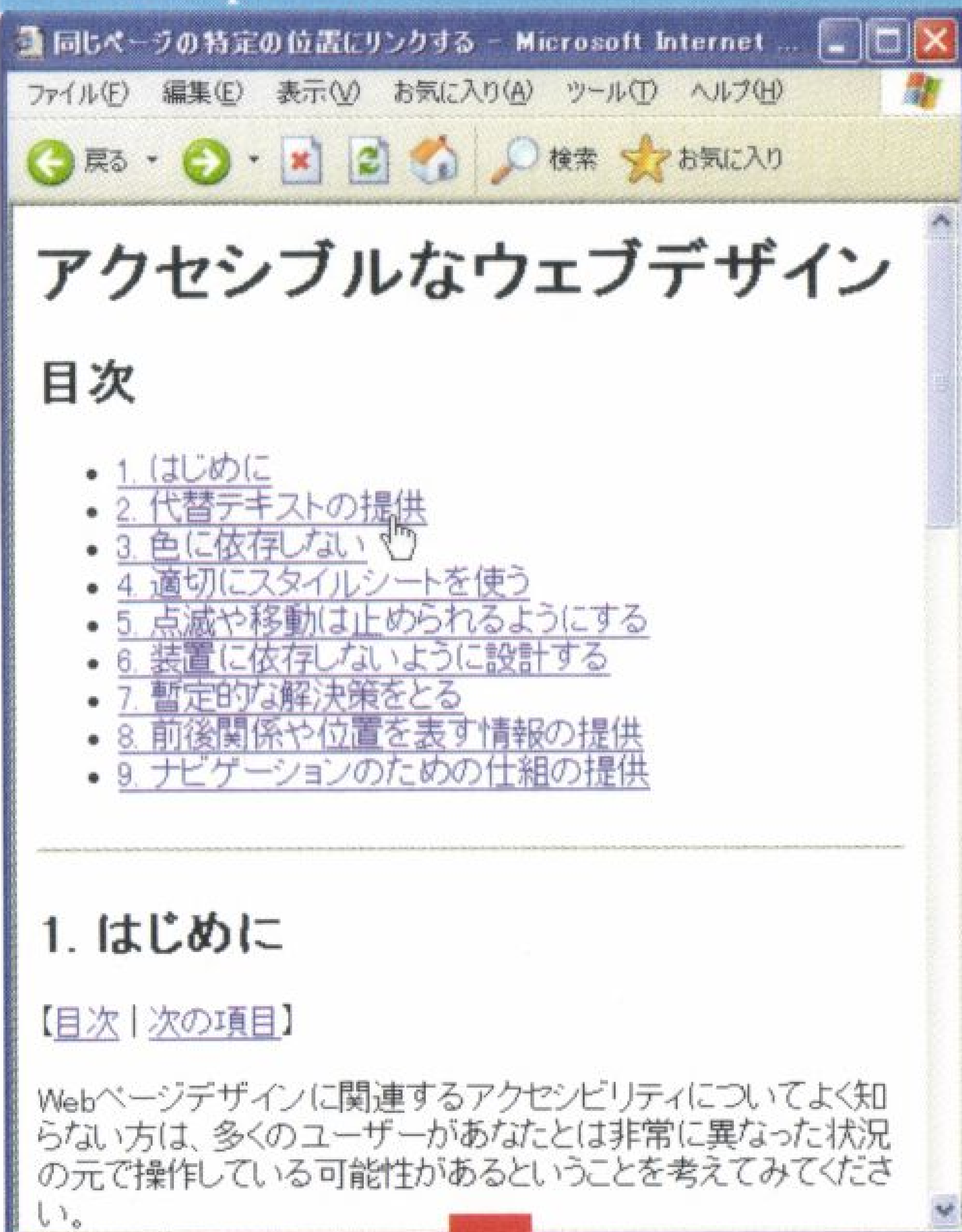
```
<p>
<a href="index.html">ホーム</a> |
<a href="wtnew.html">更新情報</a> |
<a href="about.html">会社概要</a> |
<a href="prdct.html">製品情報</a> |
<a href="recrt.html">採用情報</a>
</p>
<p>
<a href="index.html">
<img src="icon.gif" width="88" height="31" alt="A-HREFホーム">
</a>
<br>
Copyright &copy; 2001 - 2005 A-HREF,Inc.
</p>
```


<body link="～">：「リンク」の「リンク部分の文字色を設定する」(P.59)
<img src="～">：「画像とマルチメディア」の「画像の外枠を設定する」(P.100)
スタイルシート：「CSSを適用する対象」の「リンク部分に適用させる」(P.195)

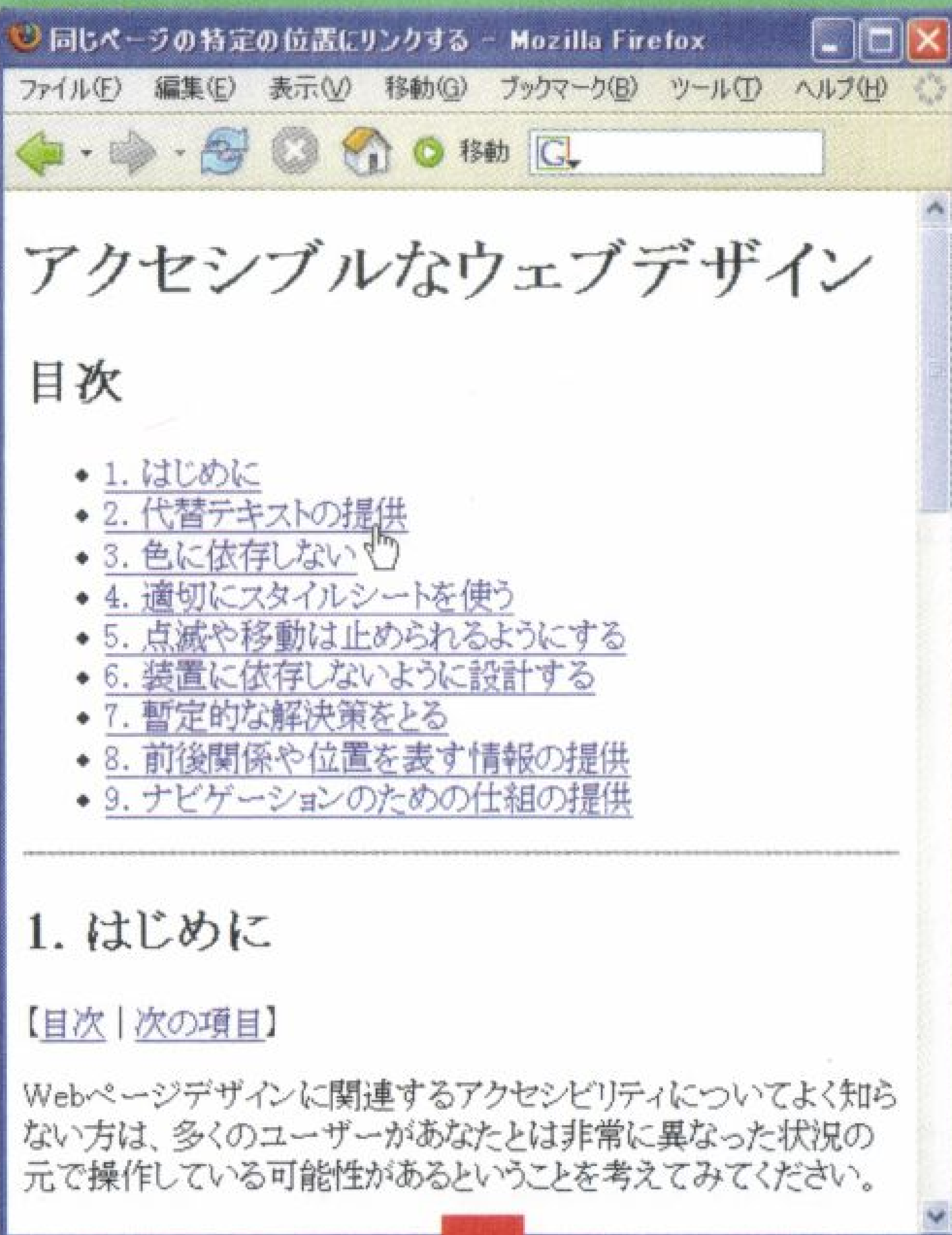
# 同じページの特定の位置にリンクする

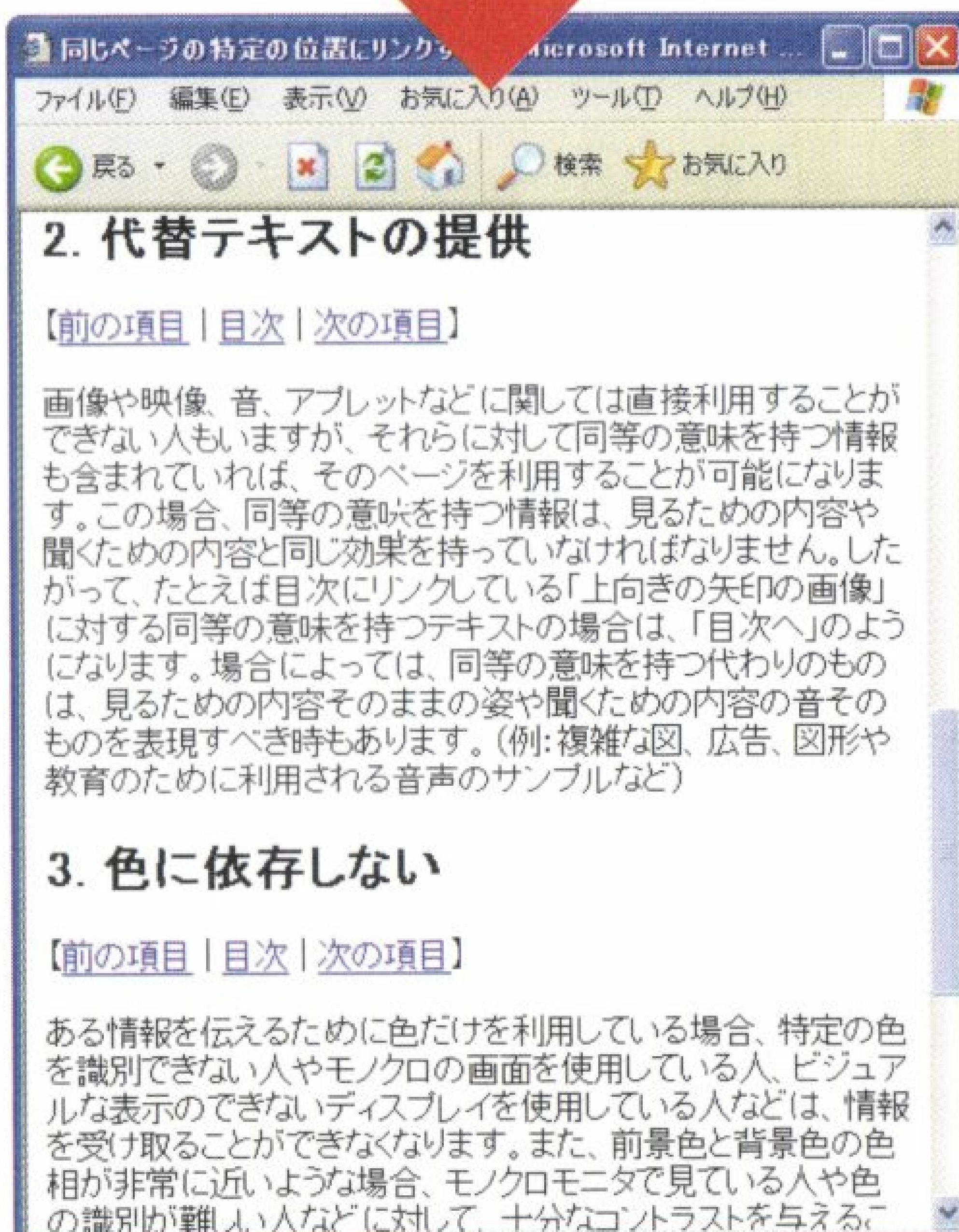
<a href="#位置名">～</a>　◀リンク元の指定（ここから）
<a name="位置名">～</a>　◀リンク先の指定（この位置へ）

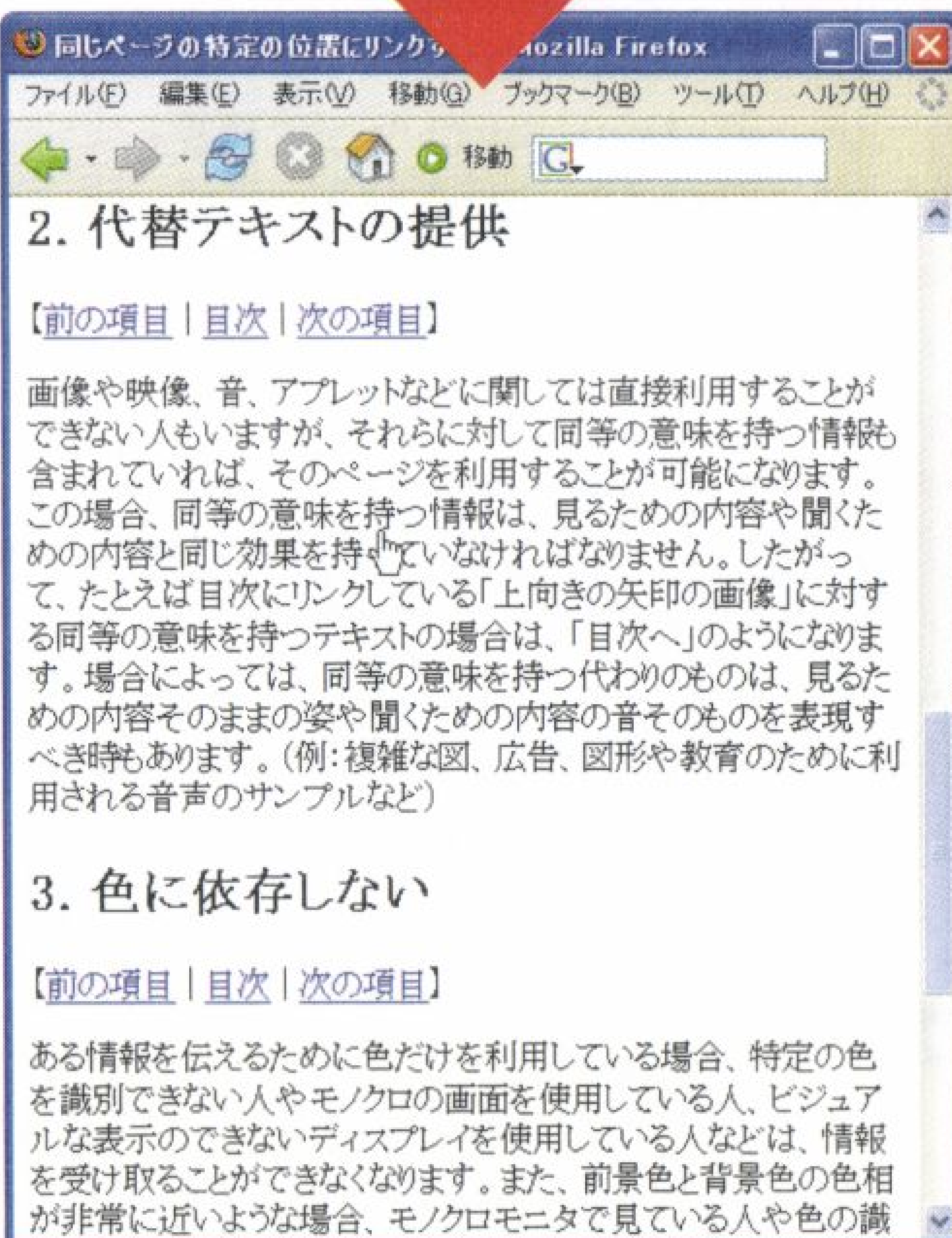
Internet Explorer 6.0




Firefox 1.0



ひとつのページがとても長い場合などに、同じページ内の特定の位置に名前を付けておいて、そこにリンク(ジャンプ)することができます。リンクの対象となる位置に名前を付けるには、name属性を使用します。そして、そこへリンクするためには、href属性でリンク先の名前の前に「#」記号を付けて指定します。

**Sample**

```
<h1>アクセシブルなウェブデザイン</h1>
<h2><a name="contents">目次</a></h2>
<ul>
<li><a href="#intro">1. はじめに</a></li>
<li><a href="#equiv">2. 代替テキストの提供</a></li>
<li><a href="#color">3. 色に依存しない</a></li>
  ∫
</ul>
<hr>

<h2><a name="intro">1. はじめに</a></h2>
<p>
【<a href="#contents">目次</a>
 | <a href="#equiv">次の項目</a>】
</p>
<p>Webページデザインに関連するアクセシビリティ…</p>

<h2><a name="equiv">2. 代替テキストの提供</a></h2>
<p>
【<a href="#intro">前の項目</a>
 | <a href="#contents">目次</a>
 | <a href="#color">次の項目</a>】
</p>
<p>画像や映像、音、アプレットなどに関しては…</p>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac i-mode

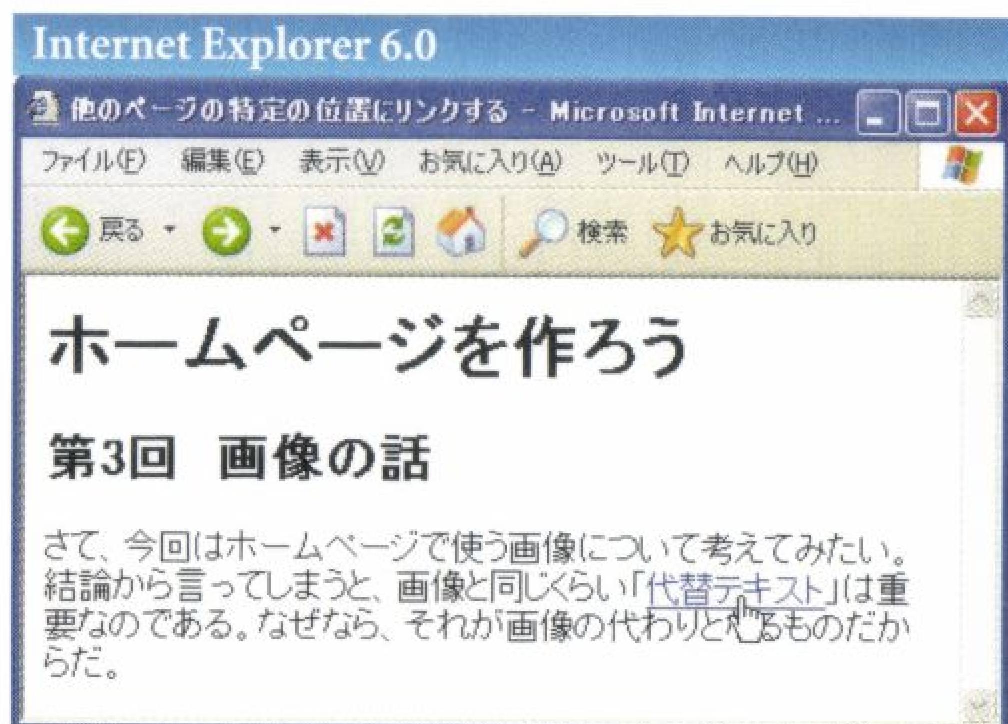
リンクする

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac i-mode

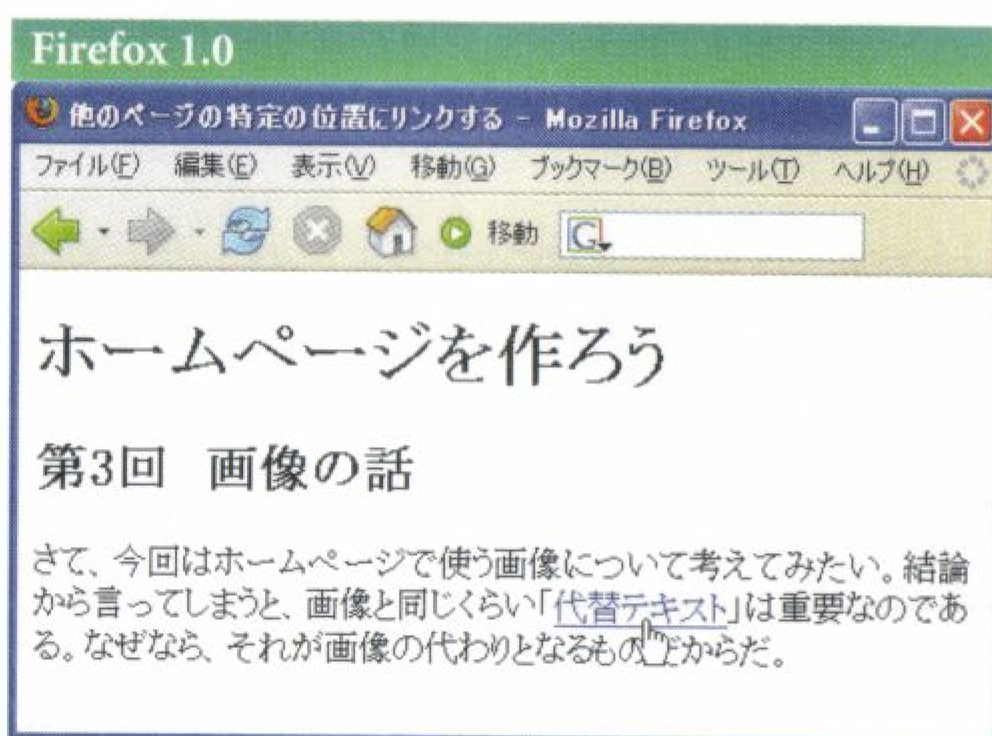
# 他のページの特定の位置にリンクする

<a href="*URL#位置名*">～</a> ⬅リンク元の指定(ここから)
<a name="*位置名*">～</a> ⬅リンク先の指定(この位置へ)

Internet Explorer 6.0




Firefox 1.0




他のページにリンクする場合に、そのページ内の特定の位置に名前を付けておいて、その位置にリンク(ジャンプ)させることができます。リンクの対象となる位置に名前を付けるには、name属性を使用します。そして、その位置へリンクするためには、href属性に「URL＋#＋位置名」を指定します。

## Sample

```
<h1>ホームページを作ろう</h1>
<h2>第3回　画像の話</h2>
<p>
さて、今回はホームページで使う画像について考えてみたい。
結論から言ってしまうと、画像と同じくらい「<a href="../
accessibility/index.html#equiv">代替テキスト</a>」は重要
なのである。なぜなら、それが画像の代わりとなるものだからだ。
  ⁓
</p>
```

※リンク先のソースについては、「同じページの特定の位置にリンクする」(P.56)を参照してください。

リンクする

# リンク部分の文字色を設定する

**<body link="色指定" vlink="色指定" alink="色指定">～</body>**

| | |
|---|---|
| link | まだ見ていないリンク部分の文字色 |
| vlink | すでに見たリンク部分の文字色 |
| alink | マウスボタンを押した時のリンク部分の文字色 |

Internet Explorer 6.0




Firefox 1.0




body要素で、ページ全体を通してのリンク部分の文字色を設定します。
まだ見ていない(キャッシュされていない)リンク、すでに見た(キャッシュされている)リンク、リンクの上でマウスボタンを押した時の3種類の状態の色を指定することができます。これらの色を設定する場合には、同時にページ全体を通しての文字色と背景色も設定しておくようにしてください。そうしないと、ユーザーがブラウザの設定で背景色を変更している場合などに、文字色と背景色が同じ系統の色になってしまい、文字が読めなくなってしまう可能性があります。
なお、HTMLで色を設定する方法は、さまざまな理由から非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。色を設定する場合には、できるだけスタイルシートを利用するようにしてください。

**Sample**

```
<body text="#000000" bgcolor="#ffffff"
 link="#339900" vlink="#666666" alink="#ff9900">
  ～
<div id="footer">
<p>
<a href="index.html">ホーム</a> |
<a href="wtnew.html">更新情報</a> |
<a href="about.html">会社概要</a> |
<a href="prdct.html">製品情報</a> |
<a href="recrt.html">採用情報</a>
</p>
<p>Copyright &copy; 2001 - 2005 LinkColor,Inc.</p>
</div>
</body>
```

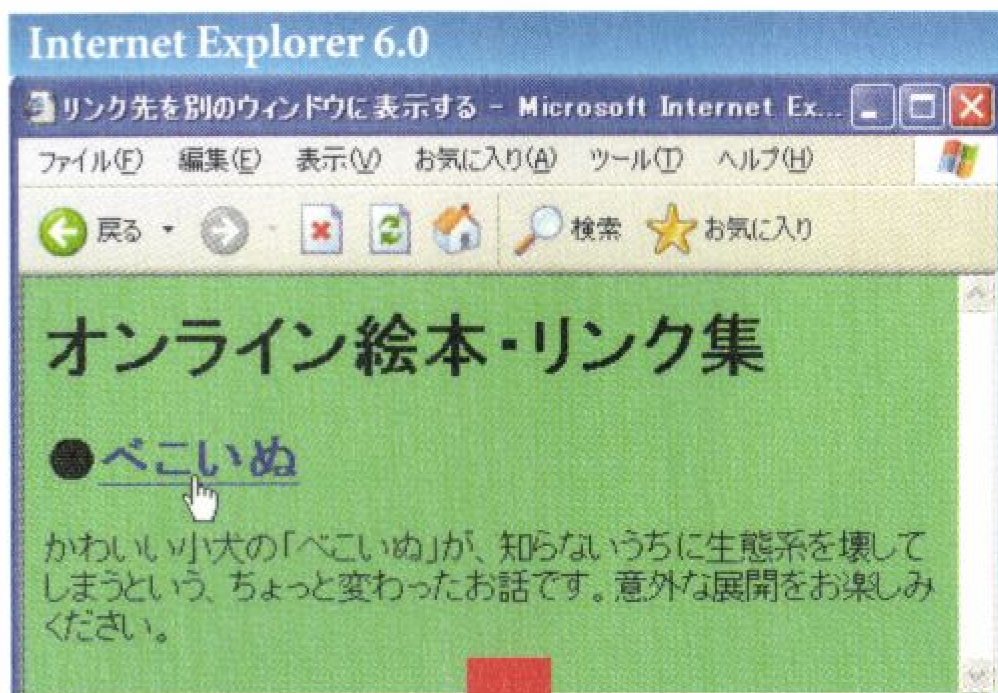

色指定：「HTMLについて」の「色の指定方法」(P.6)
<body text="～">：「基本的な内容」の「全体の文字色を設定する」(P.12)
<body bgcolor="～">：「基本的な内容」の「全体の背景色を設定する」(P.13)
スタイルシート：「CSSを適用する対象」の「リンク部分に適用させる」(P.195)
色指定：巻末付録「カラーチャート1～3」(巻末)

# リンク先を別のウィンドウに表示する

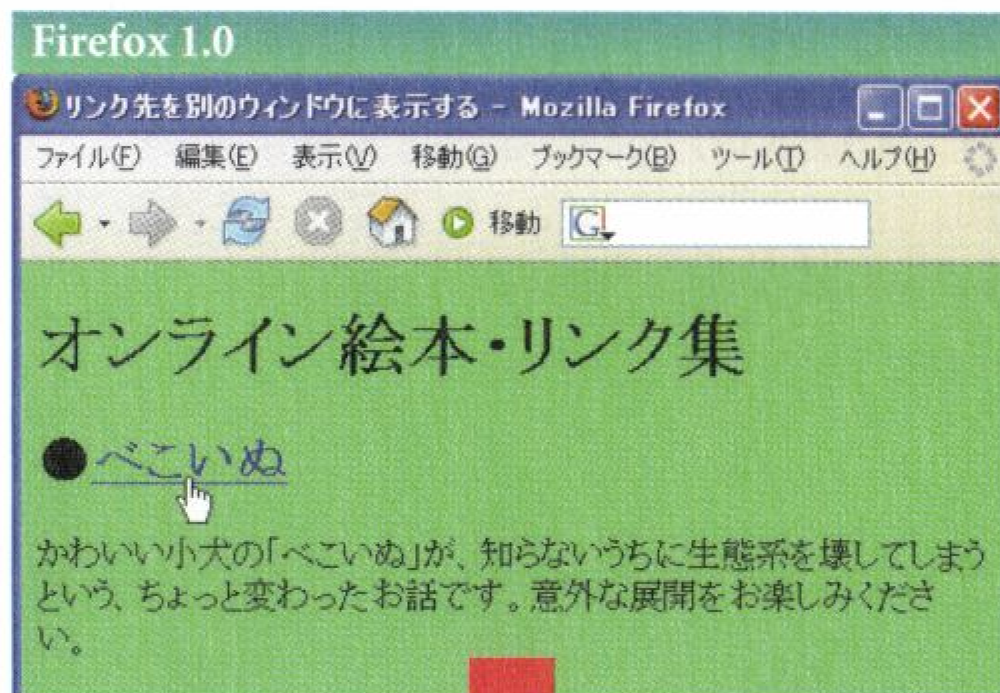
**<a href="*URL*" target="ウィンドウ名">～</a>**



Internet Explorer 6.0

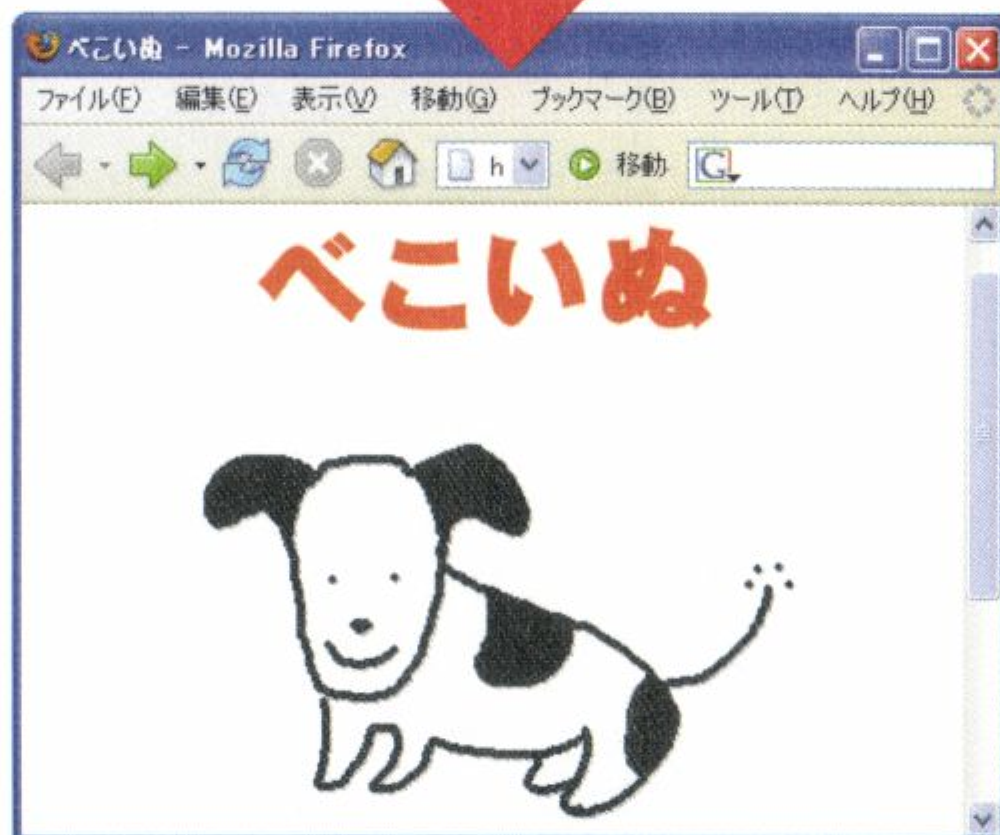



Firefox 1.0










target属性を利用すると、リンク先を表示させるウィンドウを指定することができます。指定した名前のウィンドウがすでにある場合はそのウィンドウへ、ない場合は新しいウィンドウを開いて表示します。新しく開いたウィンドウの名前は、target属性で指定したものになります。また、「_blank」のように「_」で始まる特別な名前も用意されています。名前として「_blank」を指定すると名前のない状態で新しいウィンドウを開いて表示し、「_self」を指定するとリンク元と同じウィンドウに表示します。

ただし、ユーザーの環境によっては(全画面で見ている場合や、音声ブラウザなどを利用している場合)、新しいウィンドウが開いたことがわからずに混乱する場合もありますし、リンク先を同じウィンドウに表示させたいと思うユーザーもいます。そのような意味で、リンク先を別のウィンドウに表示させることは、通常は避けたほうがよいでしょう。

**Sample**

```
<h1>オンライン絵本·リンク集</h1>
<h2>●
<a href="http://www.phoenix-c.or.jp/~zspc/bekoinu/"
 target="ehon">べこいぬ</a>
</h2>
<p>
かわいい小犬の「べこいぬ」が、知らないうちに生態系を
壊してしまうという、ちょっと変わったお話です。意外な
展開をお楽しみください。
</p>
```





## TIPS

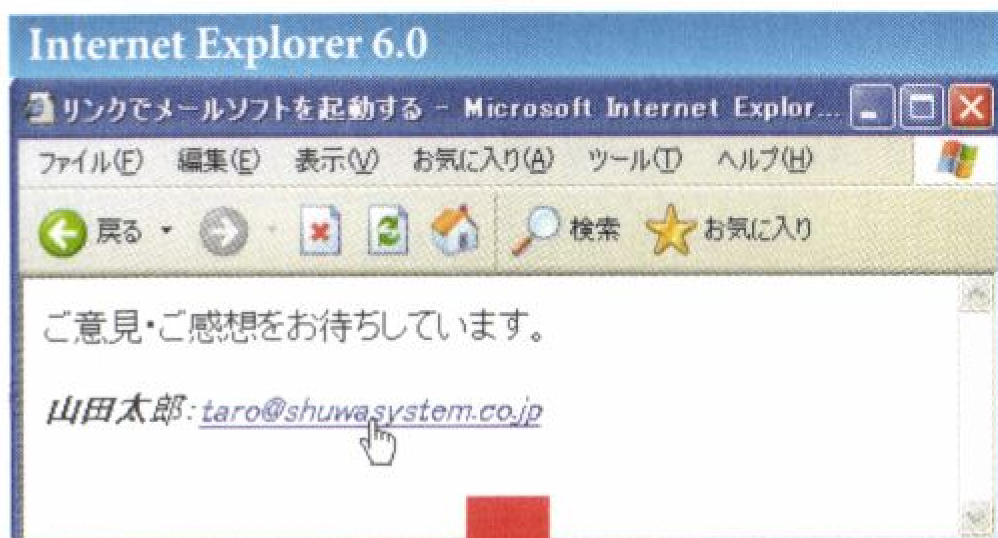
### target属性の特別な４つの値

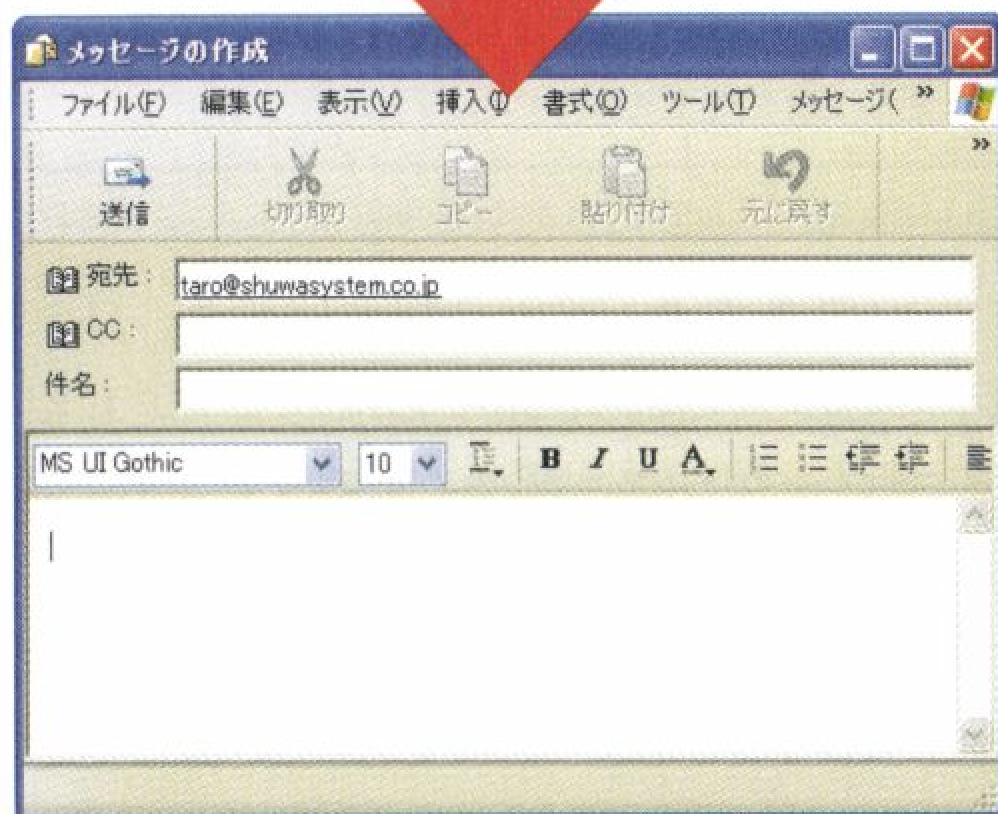
リンク先を表示するフレームやウィンドウをtarget属性で指定する場合、あらかじめ決められている４種類の特別な名前があります。それぞれの名前と機能は、次の通りです。

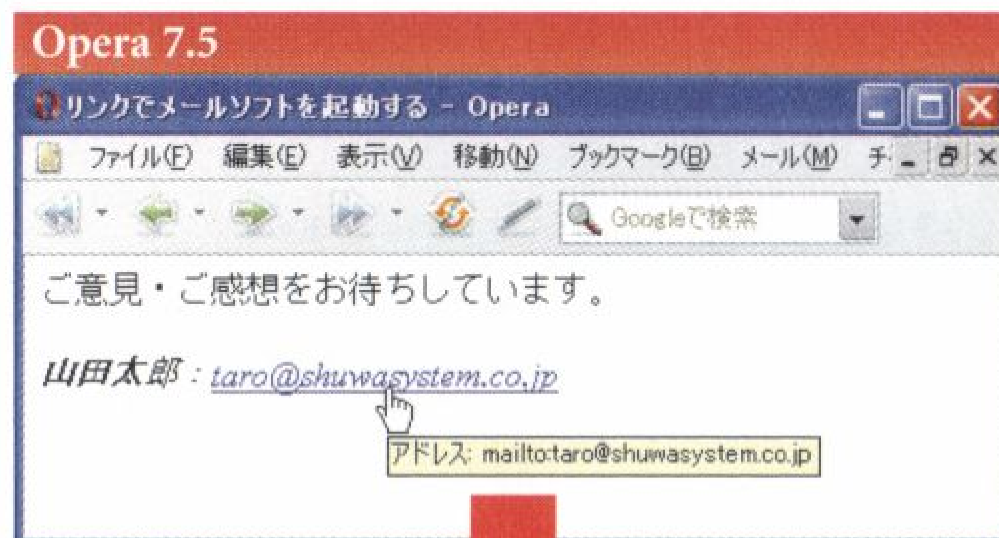
| 値 | 概要 |
|---|---|
| _top | フレームを解除して、リンク先をウィンドウ全体に表示 |
| _parent | リンク先をリンク元のフレームの親フレームに表示。親フレームがない場合には、「_self」と同様の結果になる |
| _self | リンク先をリンク元と同じフレームに表示 |
| _blank | 新しいウインドウを開いて、リンク先をそのウィンドウに表示。そのウィンドウは、名前のない状態になる |

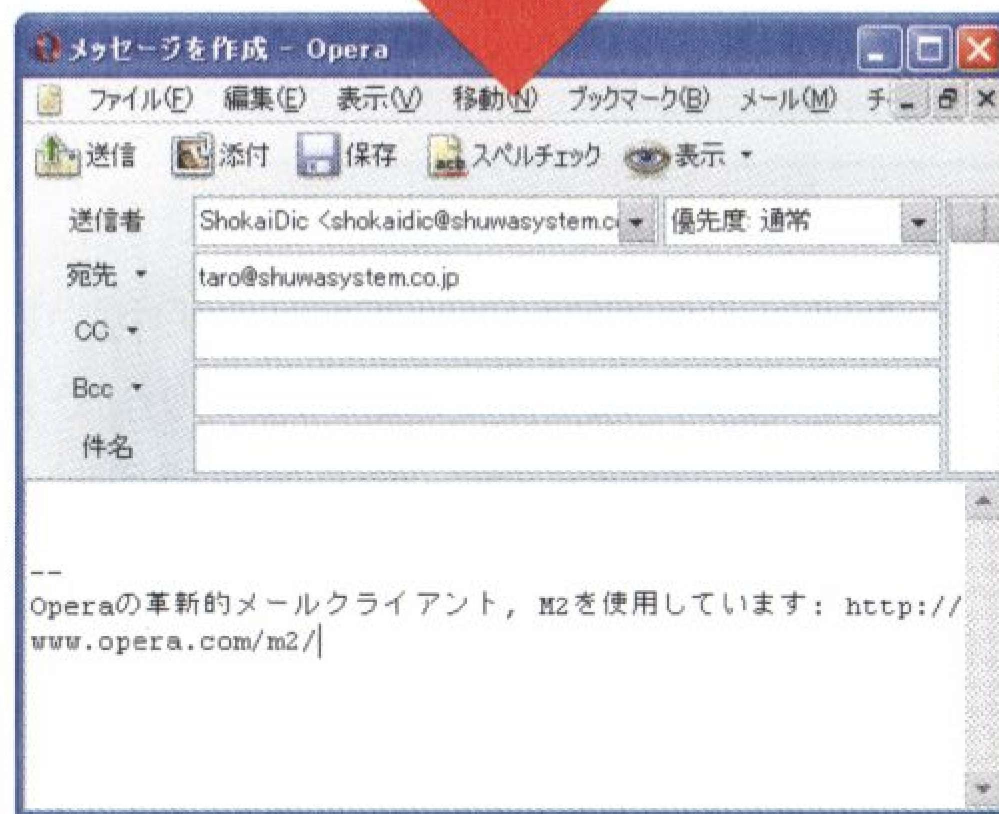
# リンクでメールソフトを起動する

## <a href="mailto:メールアドレス">～</a>




リンク部分をクリックすることで、自動的にメールソフトが起動するようにします。リンクのURLを記入する部分に、「mailto:」に続けてメールアドレスを記入してください。一般的な環境では、指定したメールアドレスが入力された状態で、メールソフトが起動します。
ただし、この機能は必ずしもすべてのブラウザで利用できるわけではありません。メールアドレスは、必ず画面に表示されるように(リンク部分の文字にメールアドレスを含めるように)しておいてください。

### Sample

```
<p>
ご意見･ご感想をお待ちしています。
</p>
<address>
山田太郎:
<a href="mailto:taro@shuwasystem.co.jp">
taro@shuwasystem.co.jp</a>
</address>
```

<address>：「テキストの種類」の「連絡先を示す」(P.30)

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac
i-mode
リンクする

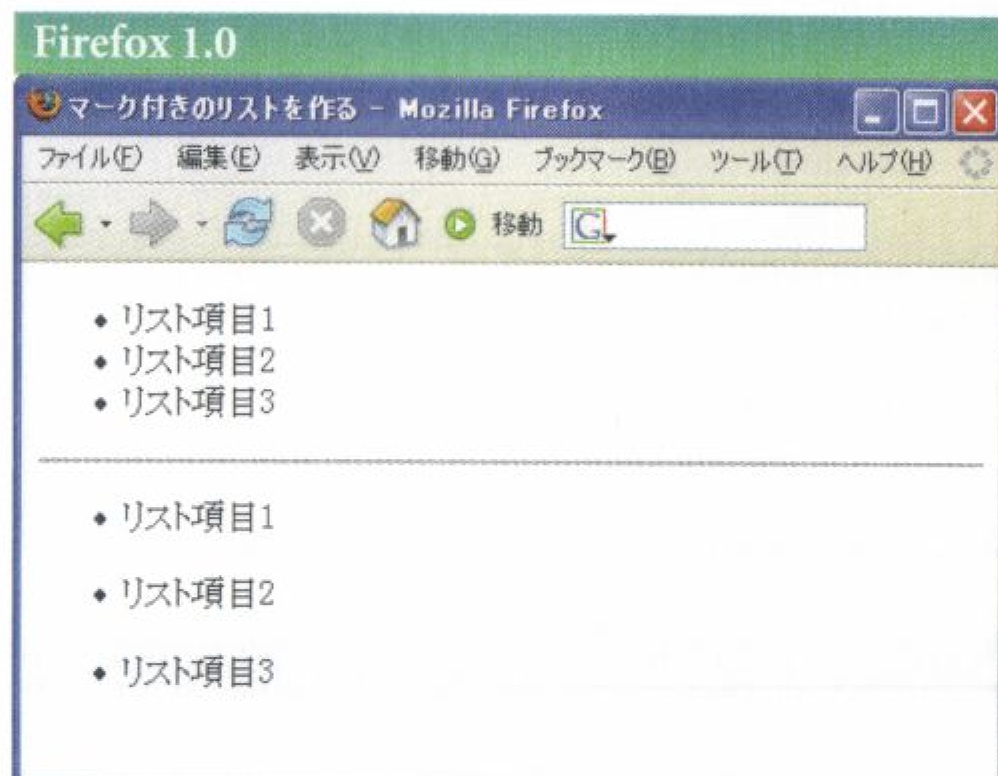
# マーク付きのリストを作る

## <ul><li>リスト項目1</li><li>リスト項目2</li>…</ul>

Internet Explorer 6.0




Firefox 1.0




ul要素は、丸や四角などのマークの付いたリスト(箇条書き)形式で表示します。リスト全体を<ul>～</ul>で囲み、その中で各項目を<li>～</li>で囲って示します。li要素の内容としては、インライン要素だけでなくブロックレベル要素(リストも含む)も入れることができます。

一般的なブラウザでは、全体にインデントされた状態で、各項目は自動的に改行されて表示されます。また、マークとしては黒丸が一般的ですが、すべてのブラウザでそのように表示されるとは限りません。

### Sample

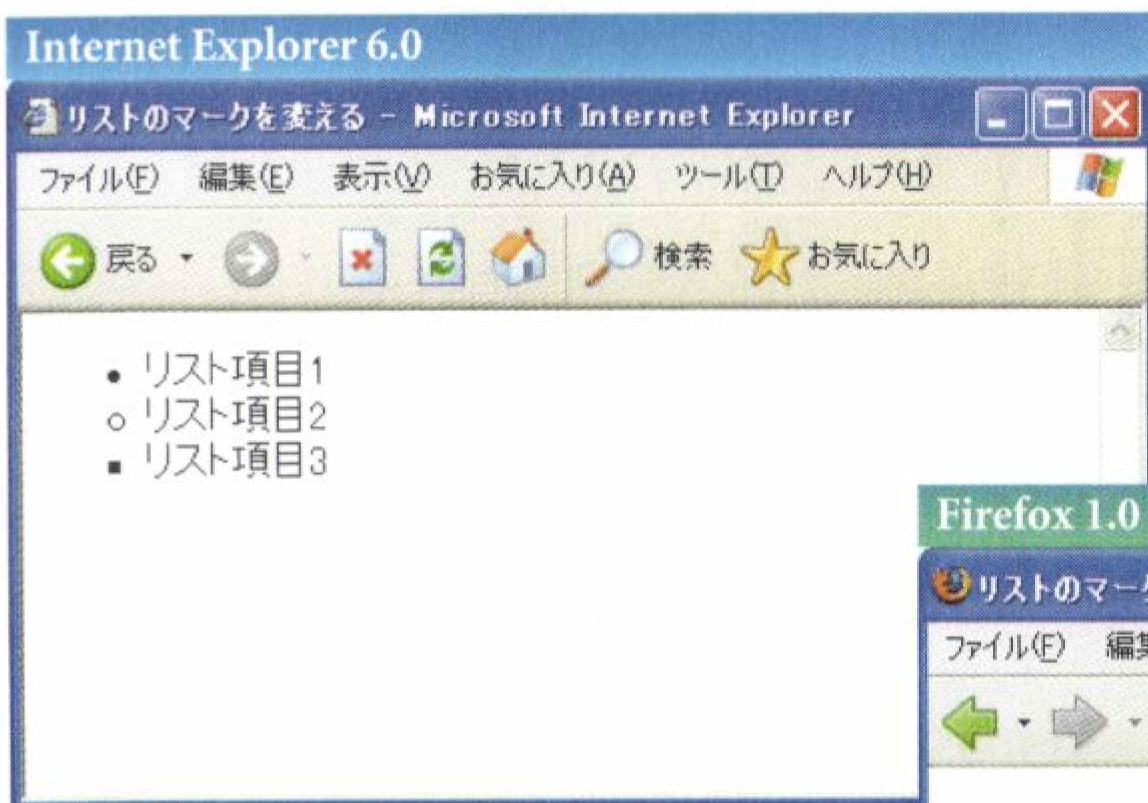
```
<ul>
<li>リスト項目1</li>
<li>リスト項目2</li>
<li>リスト項目3</li>
</ul>
<hr>
<ul>
<li><p>リスト項目1</p></li>
<li><p>リスト項目2</p></li>
<li><p>リスト項目3</p></li>
</ul>
```

# リストのマークを変える

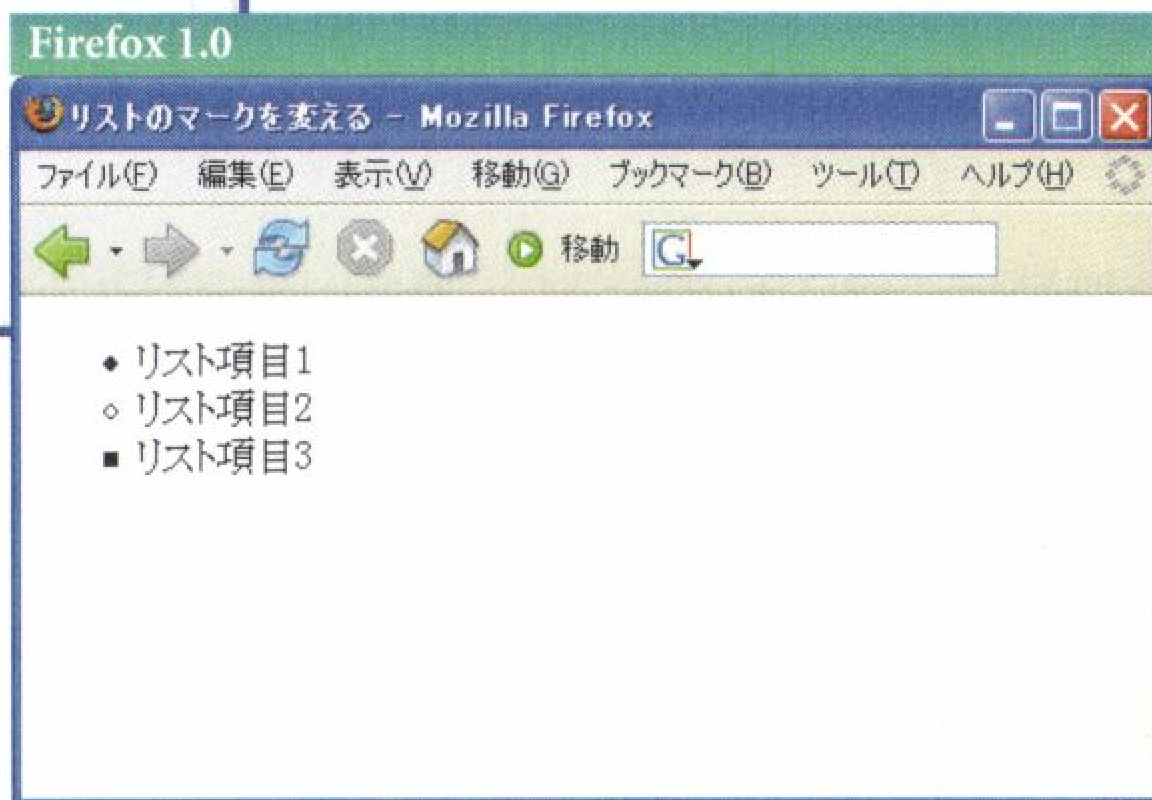
**<ul type="マークの種類">～</ul>**
**<li type="マークの種類">～</li>**

マークの種類 disc（黒丸）・circle（白丸）・square（四角）

Internet Explorer 6.0




Firefox 1.0




ul要素内のtype属性は、マーク付きリストの各項目の前に表示されるマークの種類を設定します。
<ul>に指定するとリスト全体がそのマークに変更され、<li>に指定するとその項目だけが変更されます。ただし、これらのマークはブラウザの種類によって、その大きさや色（squareには白抜きのものと塗りつぶしのものがあります）などが異なります。
また、この属性を使うことは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。マークの種類を設定する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<ul>
<li type="disc">リスト項目1</li>
<li type="circle">リスト項目2</li>
<li type="square">リスト項目3</li>
</ul>
```

スタイルシート：「リスト」の「リストのマークや番号の形式を変える」（P.270）

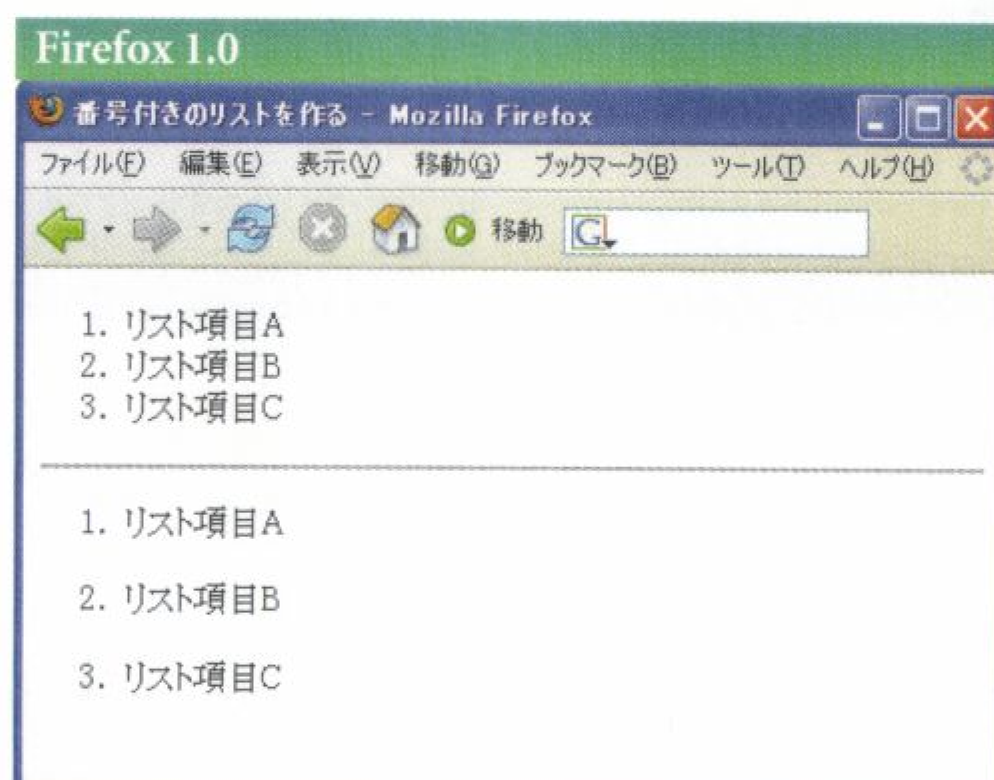
# 番号付きのリストを作る

## <ol><li>リスト項目1</li><li>リスト項目2</li>…</ol>

Internet Explorer 6.0




Firefox 1.0




ol要素は、マークではなく、番号の付けられたリスト形式で表示します。リスト全体を<ol>～</ol>で囲み、その中で各項目を<li>～</li>で囲って示します。li要素の内容としては、インライン要素だけでなくブロックレベル要素(リストも含む)も入れることができます。

一般的なブラウザでは、全体にインデントされた状態で、各項目は自動的に改行されて表示されます。

### Sample

```
<ol>
<li>リスト項目A</li>
<li>リスト項目B</li>
<li>リスト項目C</li>
</ol>
<hr>
<ol>
<li><p>リスト項目A</p></li>
<li><p>リスト項目B</p></li>
<li><p>リスト項目C</p></li>
</ol>
```

# 番号の形式を変える

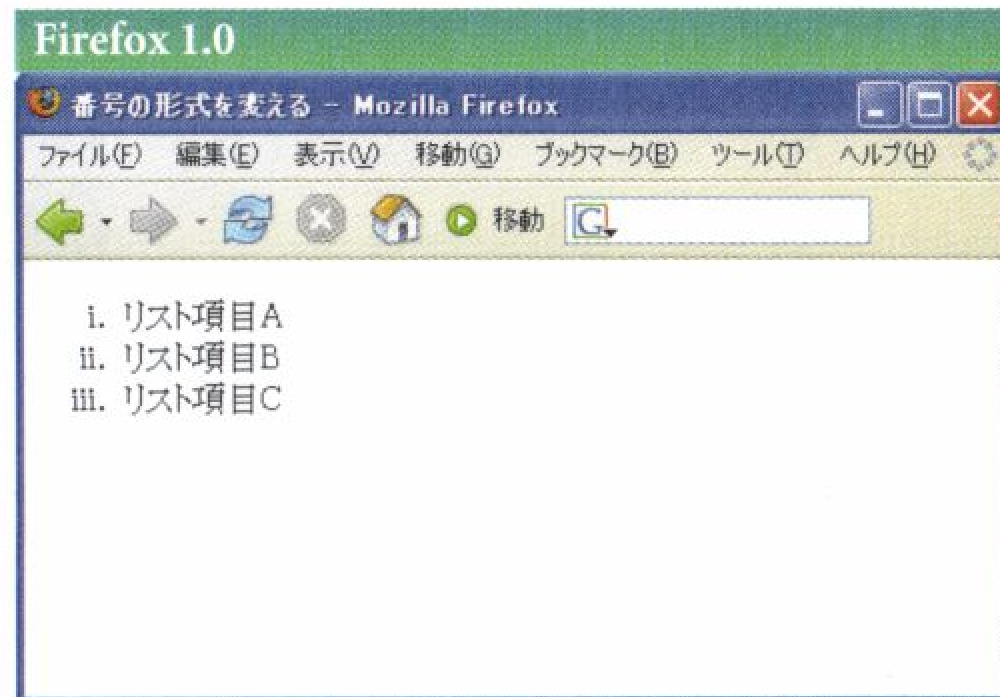
<ol type="番号の形式">～</ol>
<li type="番号の形式">～</li>

| 【番号の形式】 | |
|---|---|
| 1（算用数字） | 1, 2, 3, … |
| a（英小文字） | a, b, c, … |
| A（英大文字） | A, B, C, … |
| i（ローマ数字小文字） | i, ii, iii, … |
| I（ローマ数字大文字） | I, II, III, … |

Internet Explorer 6.0




Firefox 1.0




ol要素内のtype属性は、番号付きリストの各項目の前に表示される番号の形式を設定します。<ol>に指定するとリスト全体がその形式に変更され、<li>に指定するとその項目だけが変更されます。
ただし、この属性を使うことは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。番号の形式を設定する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<ol type="i">
<li>リスト項目A</li>
<li>リスト項目B</li>
<li>リスト項目C</li>
</ol>
```

スタイルシート：「リスト」の「リストのマークや番号の形式を変える」（P.270）

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac i-mode

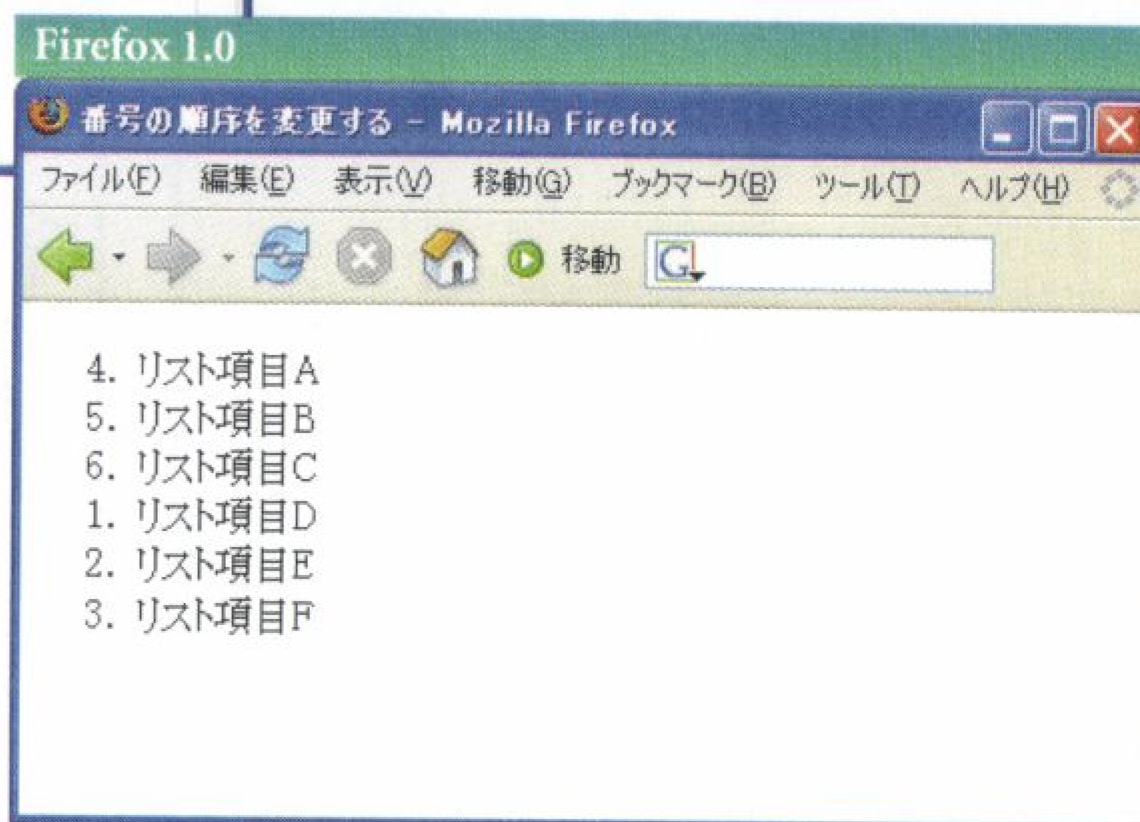
# 番号の順序を変更する

**<ol start="開始番号">～</ol>**
**<li value="開始番号">～</li>**

Internet Explorer 6.0




Firefox 1.0




ol要素にstart属性を指定すると、番号付きリストを指定した番号から開始させることができます。li要素にvalue属性を指定すると、その項目の番号が変更され、以降の項目もそれに続く番号に変更されます。
ただし、この属性を使うことは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっていますので、注意してください。

## Sample

```
<ol start="4">
<li>リスト項目A</li>
<li>リスト項目B</li>
<li>リスト項目C</li>
<li value="1">リスト項目D</li>
<li>リスト項目E</li>
<li>リスト項目F</li>
</ol>
```

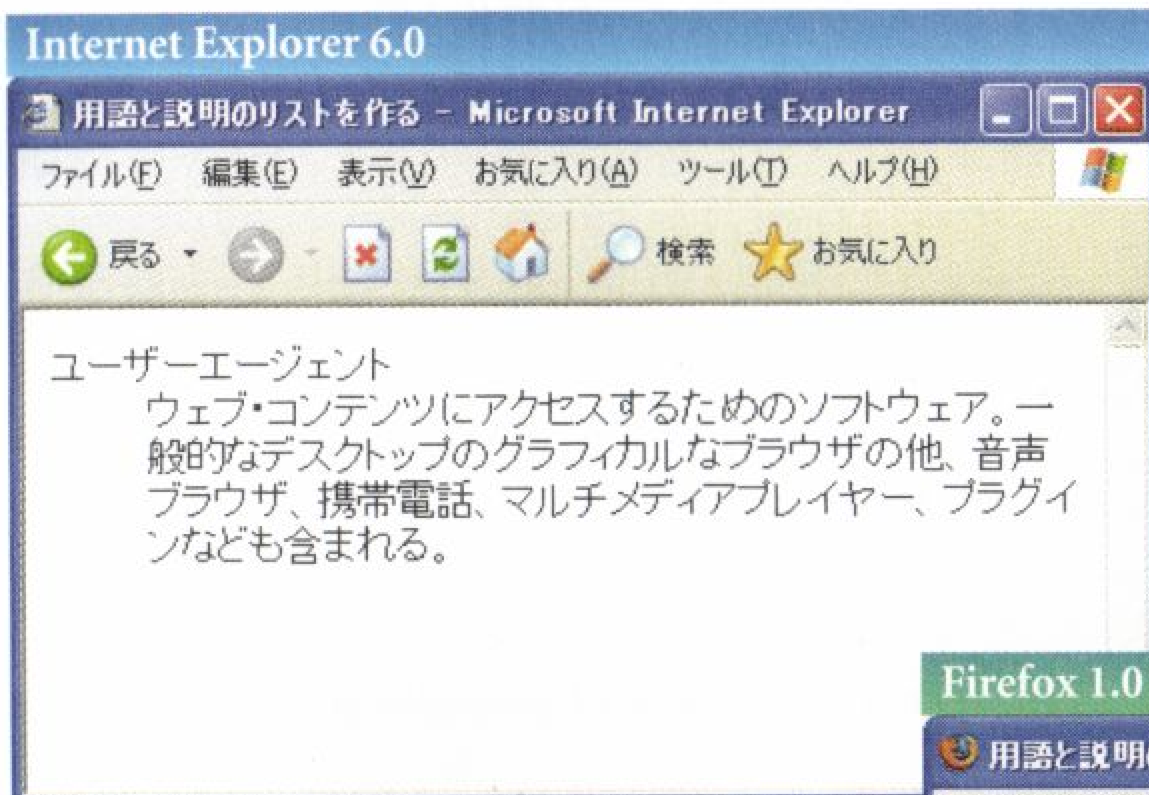
リストを作る

# 用語と説明のリストを作る

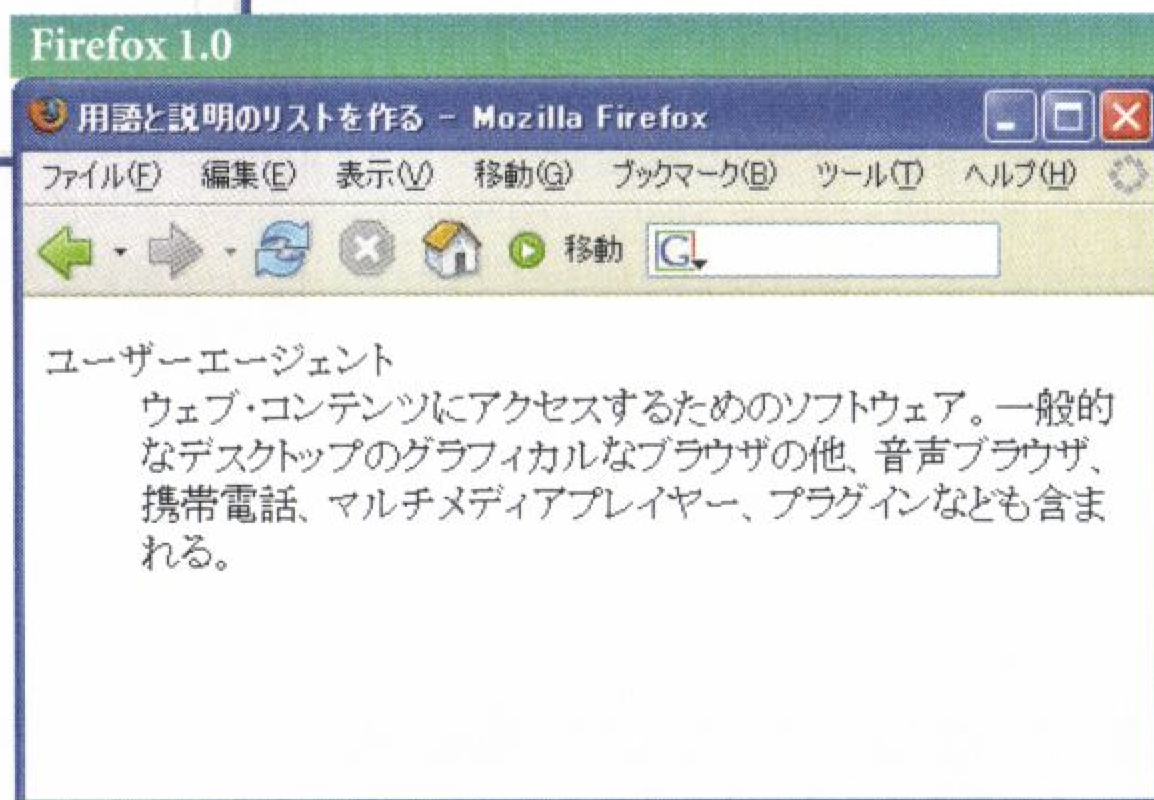
## <dl><dt>用語</dt><dd>その説明</dd>…</dl>



Internet Explorer 6.0




Firefox 1.0




用語とそれに対する説明(定義)を対にした形式で表示します。<dl>～</dl>の範囲には、dt要素(用語)とdd要素(説明)を1組だけでなく、それぞれ必要な数だけ任意の順序で入れることができます。
一般的なブラウザでは、<dd>～</dd>の範囲がインデントされて表示されます。

### Sample

```
<dl>
<dt>ユーザーエージェント</dt>
<dd>
ウェブ・コンテンツにアクセスするためのソフトウェア。
一般的なデスクトップのグラフィカルなブラウザの他、
音声ブラウザ、携帯電話、マルチメディアプレイヤー、
プラグインなども含まれる。
</dd>
</dl>
```

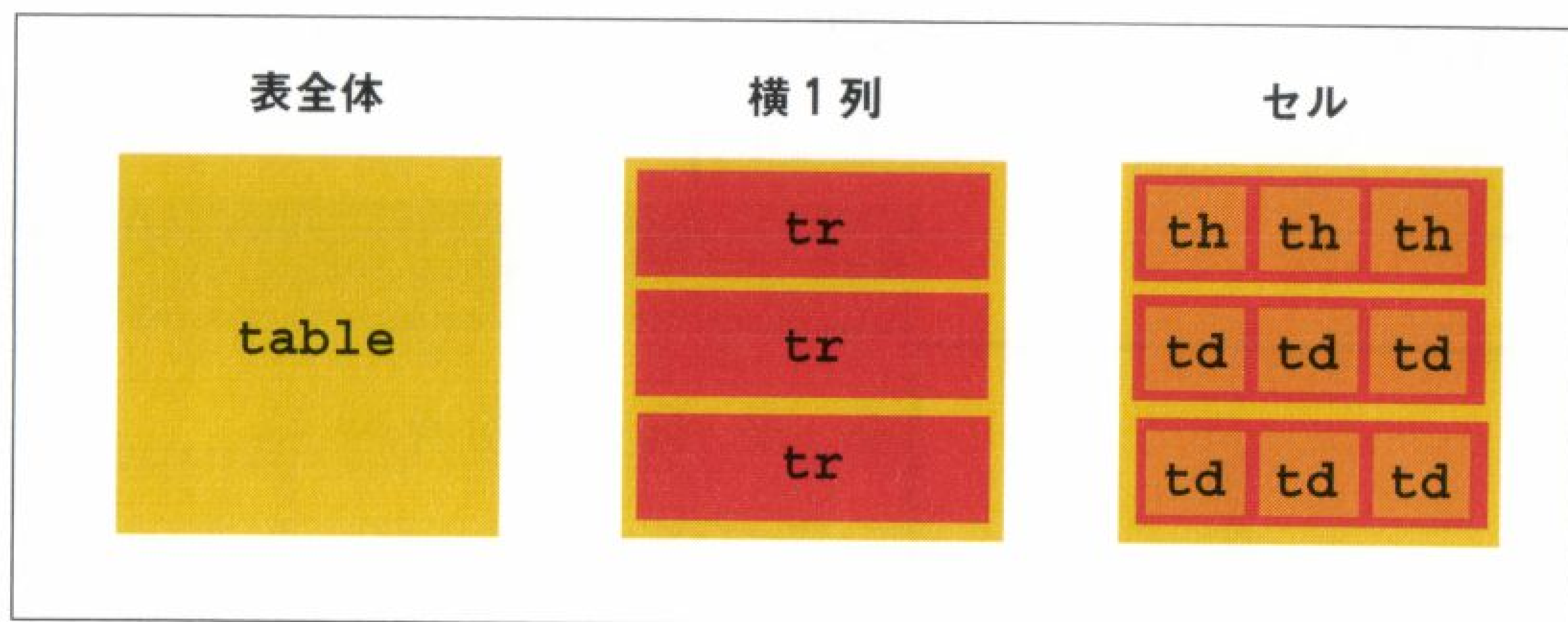
IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

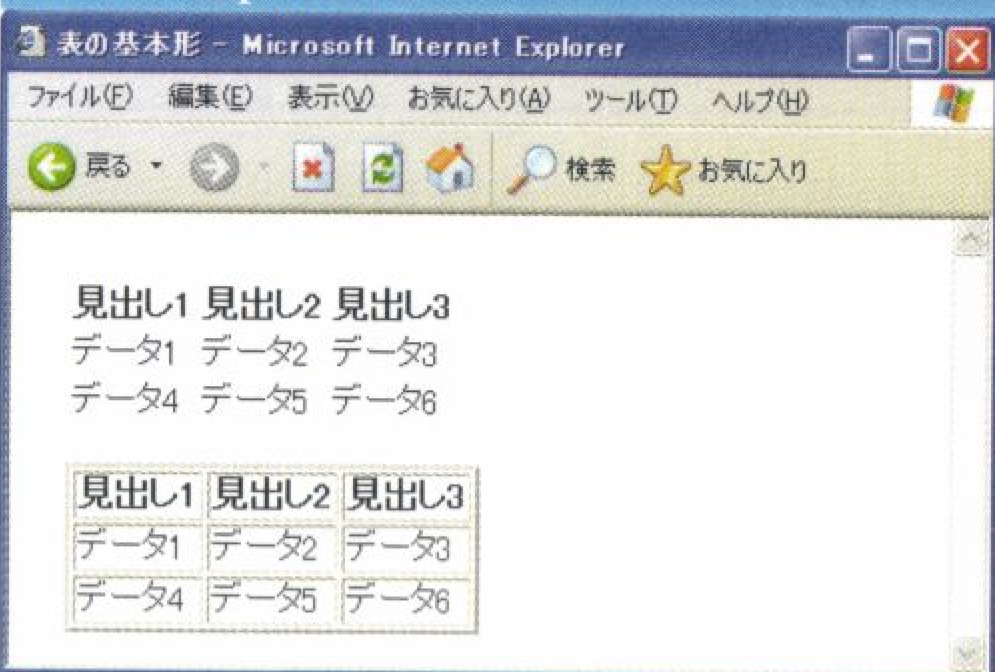
# 表の基本形

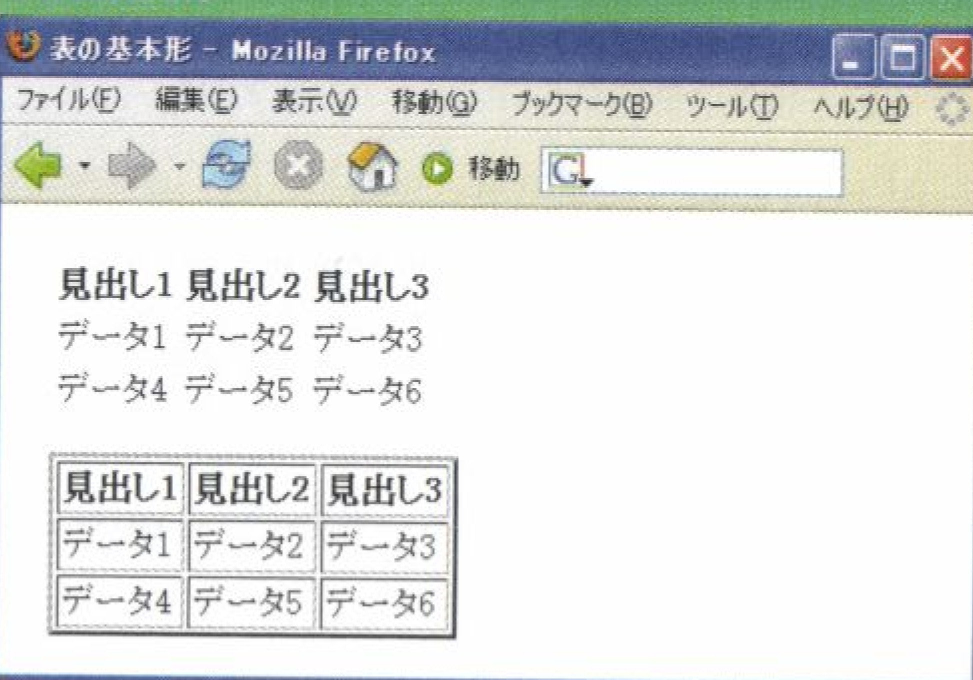
<table border="外枠の太さ">～</table> ←表全体
<tr>～</tr> ←横1列
<th>～</th> ←セル:見出し用
<td>～</td> ←セル:データ用

外枠の太さ　ピクセル

**HTMLの表の構造**




表は、その全体を<table>～</table>で囲って示します。表の中のひとつのセルは<th>～</th>、または<td>～</td>で表され、各セルは表の横1列分ずつ<tr>～</tr>で囲ってまとめます。つまり、table要素の内容は横1列を表すtr要素で構成され、各tr要素の内容はセルを表すth要素、またはtd要素で構成されるということになります。th要素は、そのセルが見出しの役割をするセルであることを表し、td要素はそのセルの内容がデータであることを示します。一般的なブラウザでは、th要素内の文字は太字でセンタリングされて表示されます。また、一般にborder属性を指定しなければ、表の枠は表示されません。

**Sample**

```
<table>
  <tr>
    <th>見出し1</th><th>見出し2</th><th>見出し3</th>
  </tr>
  <tr>
    <td>データ1</td><td>データ2</td><td>データ3</td>
  </tr>
  <tr>
    <td>データ4</td><td>データ5</td><td>データ6</td>
  </tr>
</table>

<table border="2">
  <tr>
    <th>見出し1</th><th>見出し2</th><th>見出し3</th>
  </tr>
  <tr>
    <td>データ1</td><td>データ2</td><td>データ3</td>
  </tr>
  <tr>
    <td>データ4</td><td>データ5</td><td>データ6</td>
  </tr>
</table>
```



## コラム

### 音声ブラウザでは表をどのように読み上げるのか?

HTMLのテーブルは表を表すためのものですが、実際にはレイアウトをする目的で使用されていることもあります。これは現実的には仕方がないことかもしれませんが、レイアウトの仕方によっては、音声ブラウザなどで内容がわからなくなってしまうことがありますので、注意してください。

多くの音声ブラウザは、内容を表の構造に従って、上から横1列ずつ読み上げていきます。簡単に言えば、表の内容はソースに書かれている順序で読み上げられるということです。テーブルでレイアウトをする場合には、この点に注意して、そのような順序で内容が読み上げられても、内容が理解できるようにしておいてください。

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

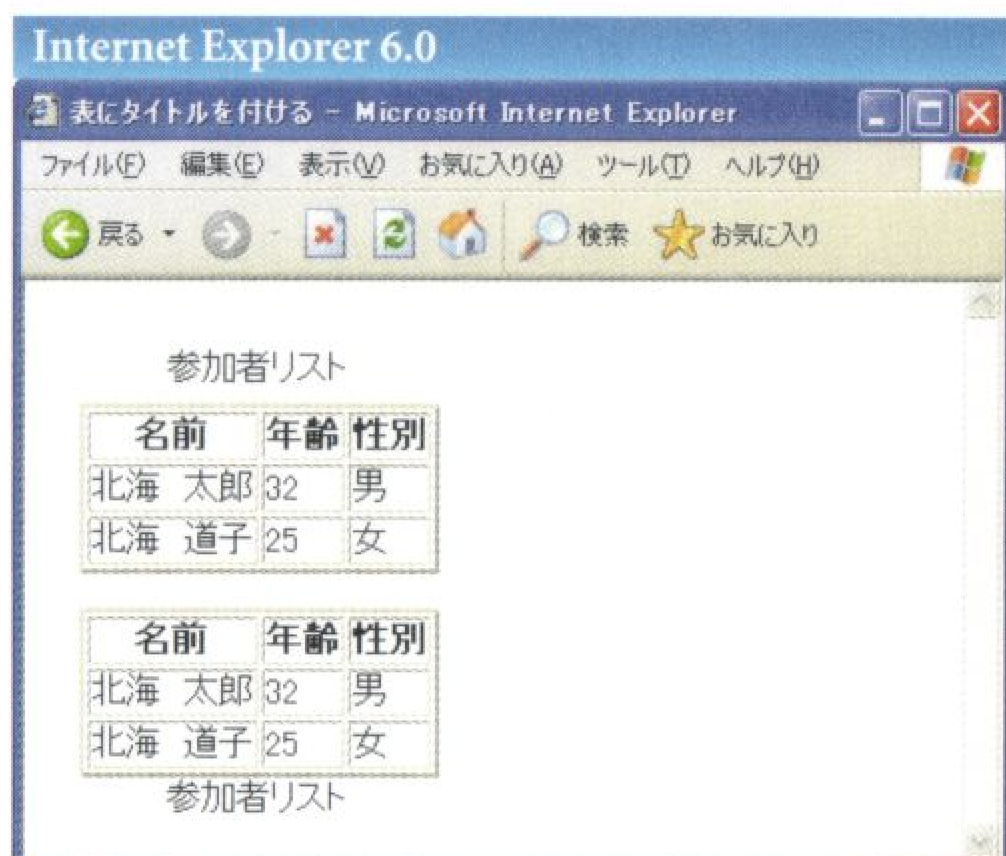
HTML

# 表にタイトルを付ける

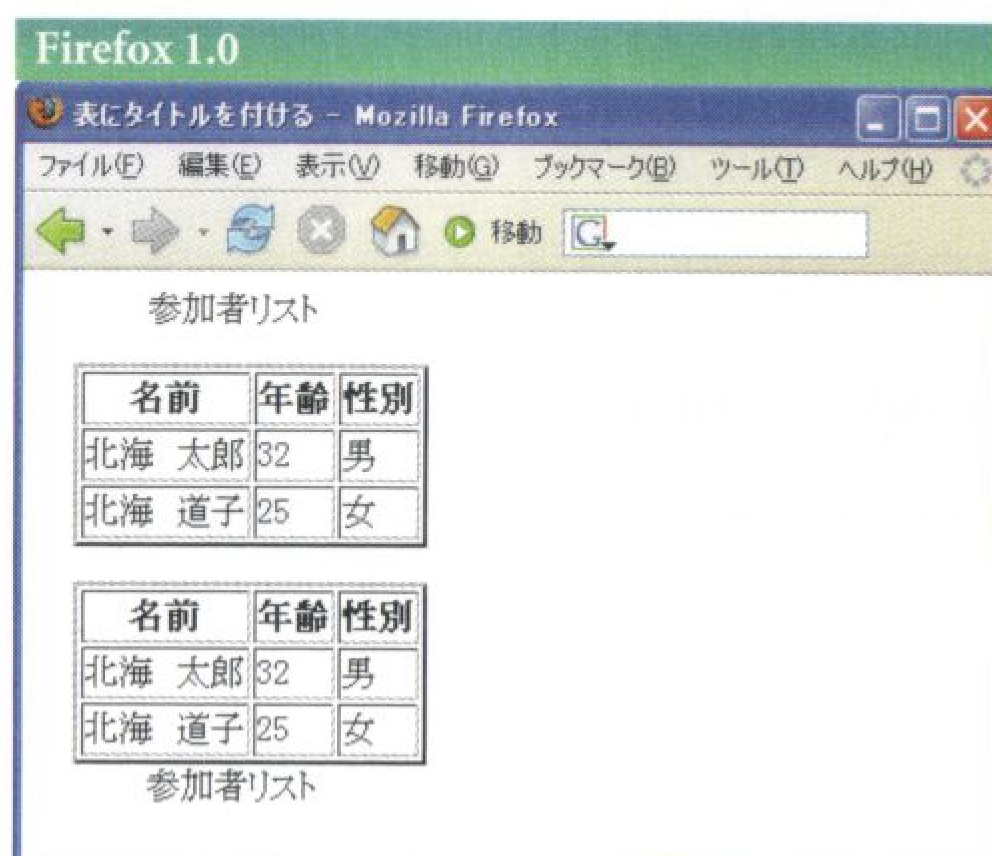
**<caption>～</caption>**
**<caption align="*表示位置*">～</caption>**

表示位置　　top・bottom

Internet Explorer 6.0




Firefox 1.0




caption要素は、表にタイトル(キャプション)を付けます。
この要素は、必ずtable要素の開始タグ<table>の直後に配置してください。表示位置としては表の上(top)、または下(bottom)が指定できますが、指定しない場合には上に表示されます。
ただし、align属性を使用することは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。表示方法を指定する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<table border="2">
<caption>参加者リスト</caption>
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>

<table border="2">
<caption align="bottom">参加者リスト</caption>
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>
```


スタイルシート：「テーブル」の「表のタイトルを下に表示させる」(P.278)

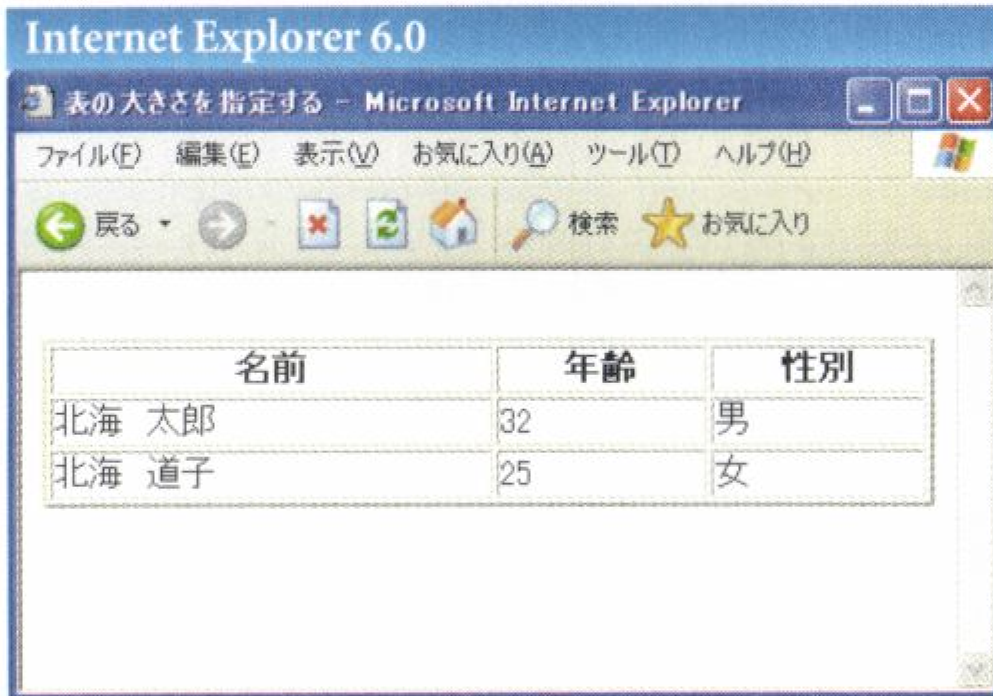
表を作る

# 表の大きさを指定する

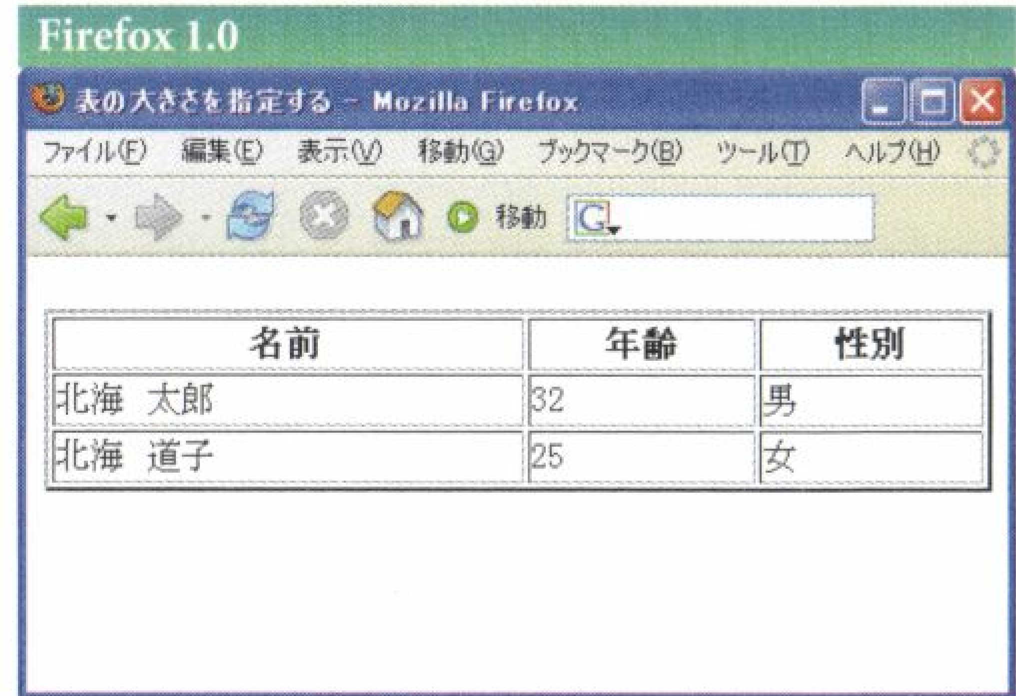
## <table width="幅">～</table>

| 幅 | ピクセル数またはウインドウに対する% |
|---|---|

Internet Explorer 6.0




Firefox 1.0




table要素のwidth属性は、表全体の幅をピクセル数または%で指定します。
一般的なブラウザの中には、高さを指定するためのheight属性をサポートしているものがありますが、標準的なHTMLの仕様ではtable要素にheight属性はありません。セルを表すth要素とtd要素にはheight属性がありますので、高さを指定したい場合はそちらを利用することもできます。

### Sample

```
<table border="2" width="100%">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>
```


<th width="～" height="～">・<td width="～" height="～">：「テーブル」の「セルの大きさを指定する」(P.74)
スタイルシート：「ボックス」の「幅と高さを指定する」(P.249)

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

# セルの大きさを指定する

**<th width="*幅*" height="*高さ*">～</th>**
**<td width="*幅*" height="*高さ*">～</td>**

| | |
|---|---|
| 幅 | ピクセル |
| 高さ | ピクセル |

Internet Explorer 6.0

Firefox 1.0

width属性はセルの幅を、height属性はセルの高さをピクセル単位で設定します。ただし、これらの属性を使用することは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。セルの大きさを指定する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<table border="2">
  <tr>
    <th width="100" height="100">名前</th>
    <th>年齢</th>
    <th>性別</th>
  </tr>
  <tr>
    <td>北海　太郎</td>
    <td>32</td>
    <td>男</td>
  </tr>
  <tr>
    <td>北海　道子</td>
    <td>25</td>
    <td>女</td>
  </tr>
</table>
```


スタイルシート：「ボックス」の「幅と高さを指定する」(P.249)

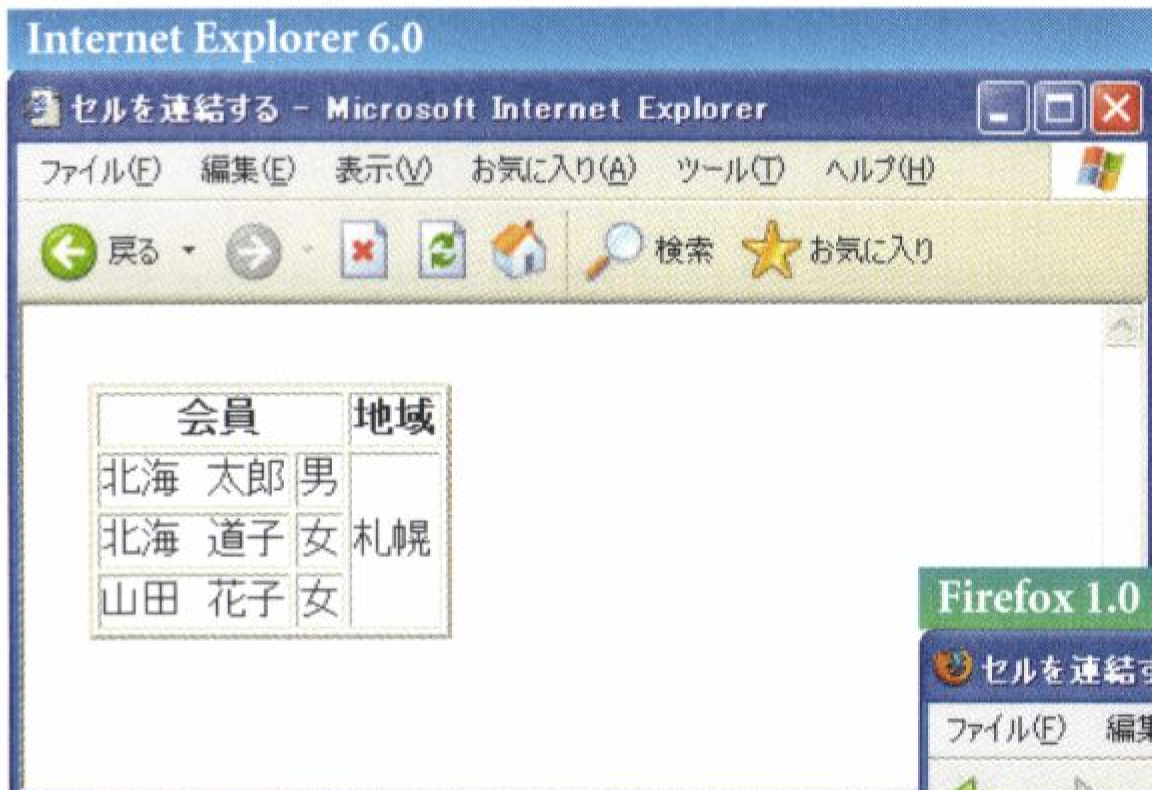
表を作る

# セルを連結する

**<th rowspan="*縦方向の連結数*">～</th>**
**<th colspan="*横方向の連結数*">～</th>**
**<td rowspan="*縦方向の連結数*">～</td>**
**<td colspan="*横方向の連結数*">～</td>**

| | |
|---|---|
| 縦方向の連結数 | そのセルから下方向へ連結するセルの数 |
| 横方向の連結数 | そのセルから右方向へ連結するセルの数 |

Internet Explorer 6.0



Firefox 1.0



rowspan属性やcolspan属性を利用すると、あるセルから指定した個数分のセルをひとつのセルとしてまとめることができます。
この時、この属性が指定されたセル(th要素またはtd要素)は、ひとつのセルで指定された個数分のセルの領域をとることになります。したがって、本来その領域内にあるはずの連結された側のセルは、ソースから取り去る必要があります。下のサンプルでは、連結されていない場合にth要素、またはtd要素があるべき位置に、コメントを配置して示しています。

## Sample

```
<table border="2">
  <tr>
    <th colspan="2">会員</th>
    <!-- ここのth要素が「会員」と連結されている -->
    <th>地域</th>
  </tr>
```

```
  <tr>
    <td>北海　太郎</td>
    <td>男</td>
    <td rowspan="3">札幌</td>
  </tr>
  <tr>
    <td>北海　道子</td>
    <td>女</td>
    <!-- ここのtd要素が「札幌」と連結されている -->
  </tr>
  <tr>
    <td>山田　花子</td>
    <td>女</td>
    <!-- ここのtd要素が「札幌」と連結されている -->
  </tr>
</table>
```



**コラム**

### セルの連結に関するHTML4.0とHTML4.01の違い

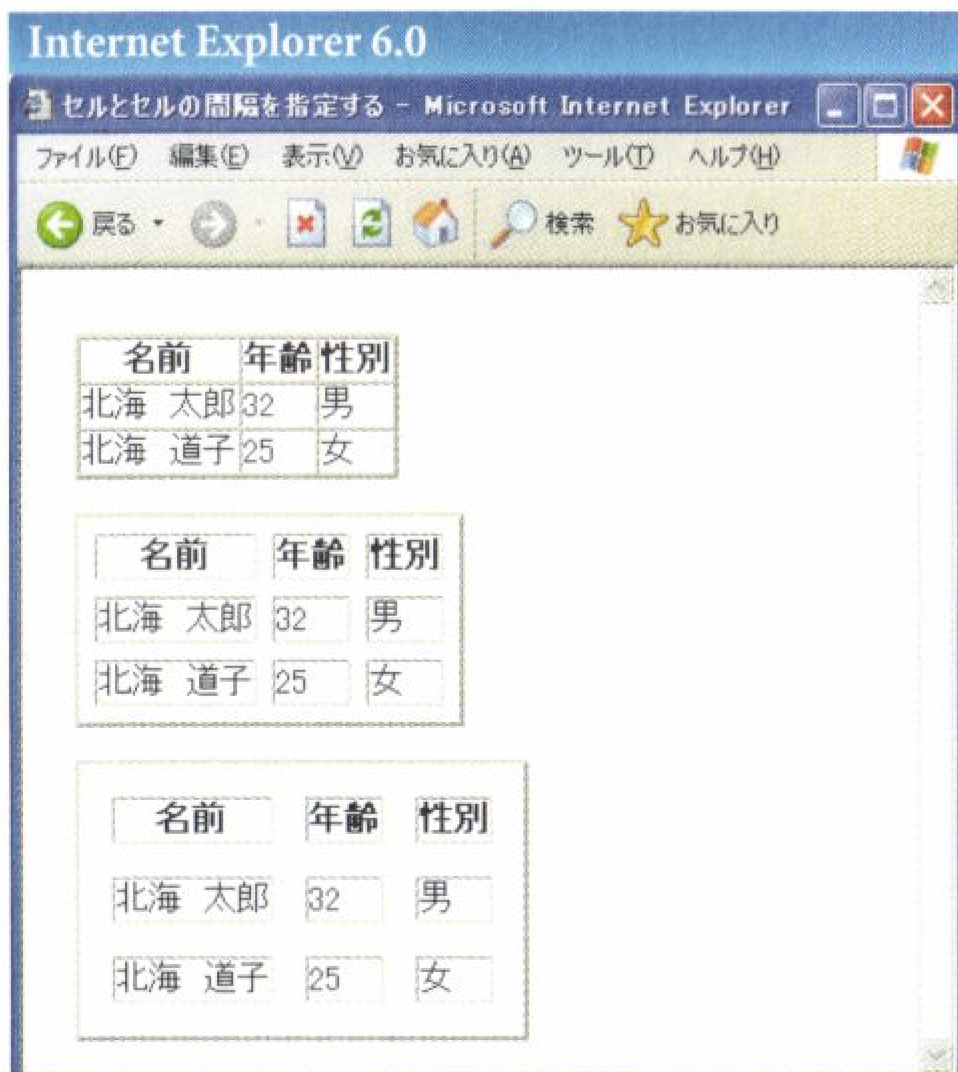
HTML4.0とHTML4.01では、rowspan属性またはcolspan属性の値に「0」を指定した場合の表示方法に関する仕様が異なっています。HTML4.0では値として「0」が指定されると、そのセル以降のすべてのセルが連結される仕様になっていますが、HTML4.01では連結される範囲がセルのグループ(thead・tbody・tfoot・colgroup)内に限定されています。現状では、値として「0」を指定しても認識しないブラウザが多いため、値として「0」ではなく、実際に連結する個数を指定したほうが安全です。

# セルとセルの間隔を指定する

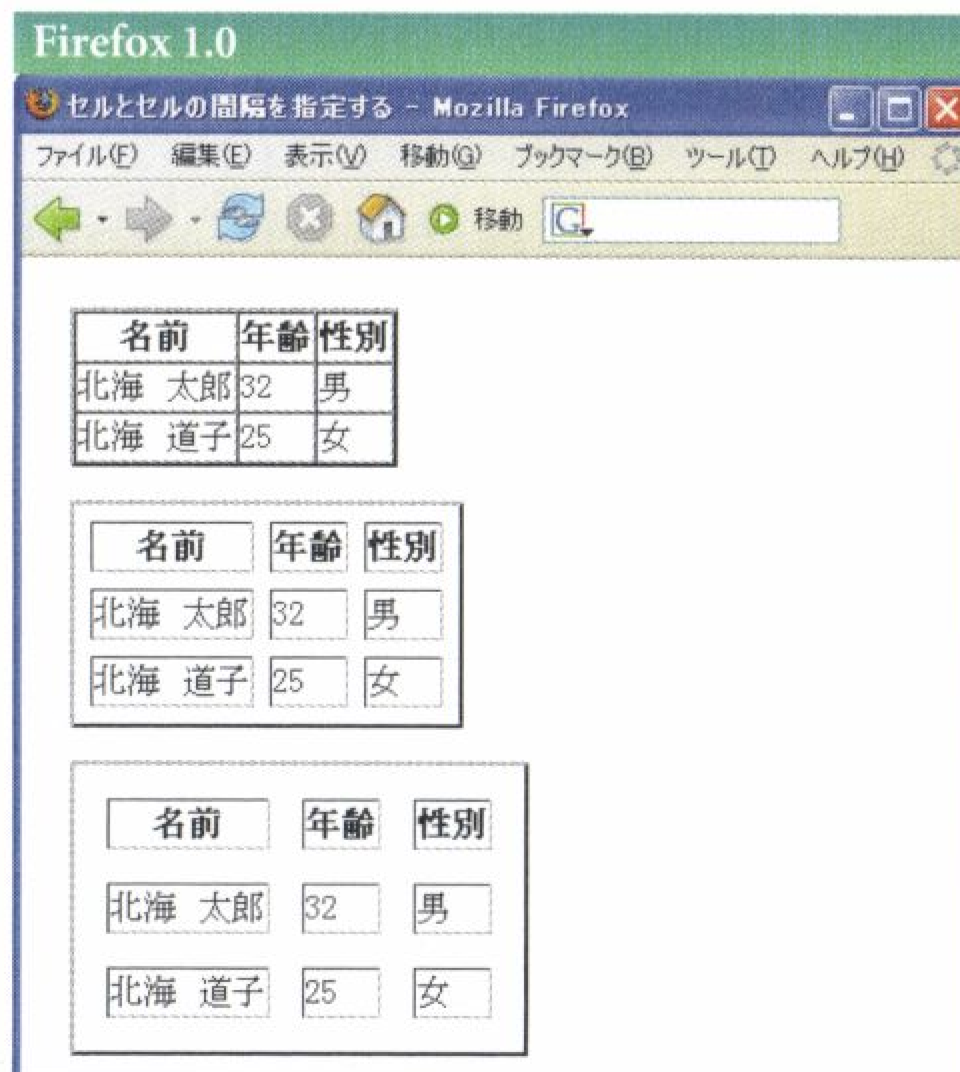
## <table cellspacing="セルの間隔">～</table>

セルの間隔　ピクセル

Internet Explorer 6.0



Firefox 1.0



cellspacing属性は、各セルとセルの間隔を設定します。
セルと表全体を囲う外枠の間も同じ間隔になります。スタイルシートにも同様の効果のあるプロパティ(border-spacing)があります。

### Sample

```
<table border="2" cellspacing="0">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>

<table border="2" cellspacing="8">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>

<table border="2" cellspacing="16">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>
```

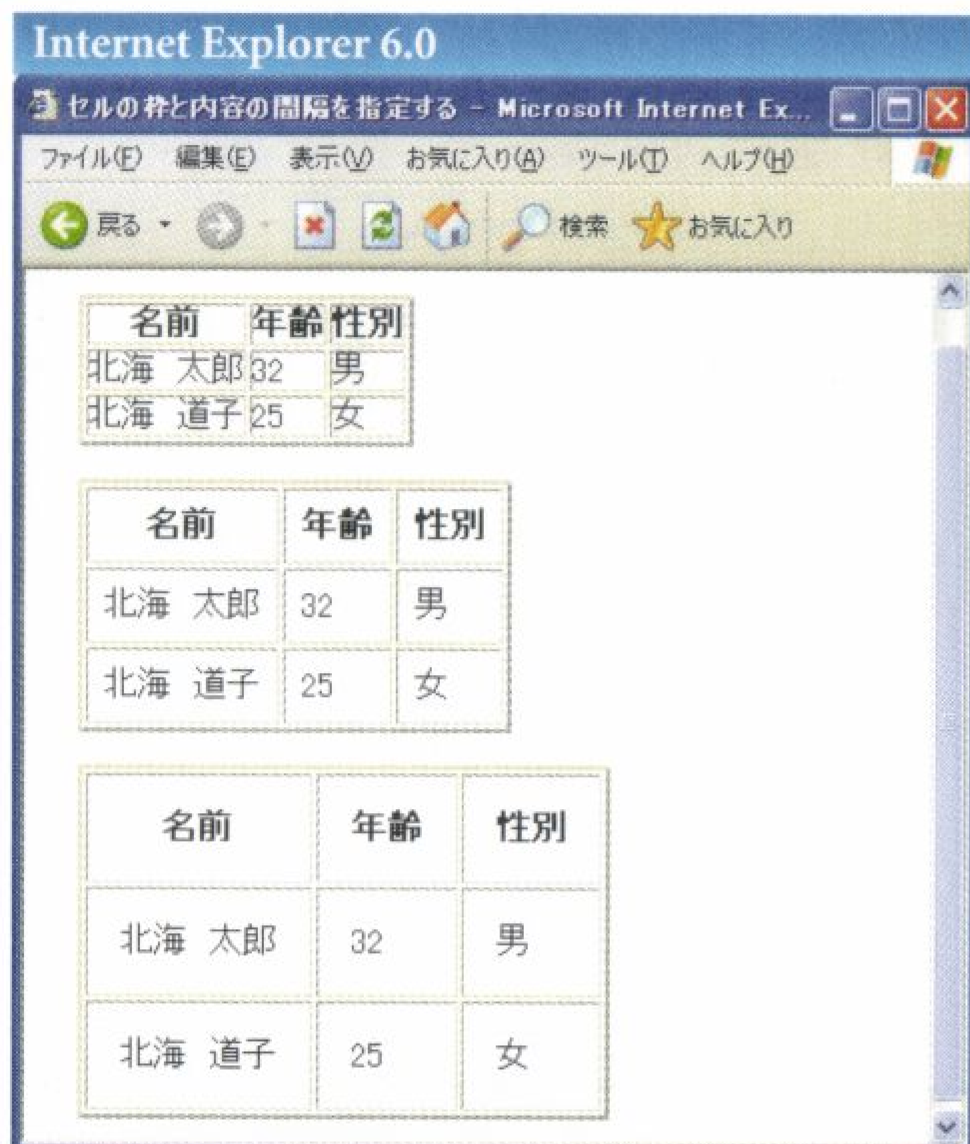

スタイルシート：「テーブル」の「セルとセルの間隔を指定する」(P.277)

# セルの枠と内容の間隔を指定する

**<table cellpadding="*セルの枠と内容の間隔*"> ~ </table>**

| セルの枠と内容の間隔 | ピクセル |
|---|---|

Internet Explorer 6.0




Firefox 1.0




cellpadding属性は、セルの枠と内容との間隔を設定します。
スタイルシートにも同様の効果のあるプロパティ(paddingなど)があります。

## Sample

```
<table border="2" cellpadding="0">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>

<table border="2" cellpadding="8">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>

<table border="2" cellpadding="16">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>
```

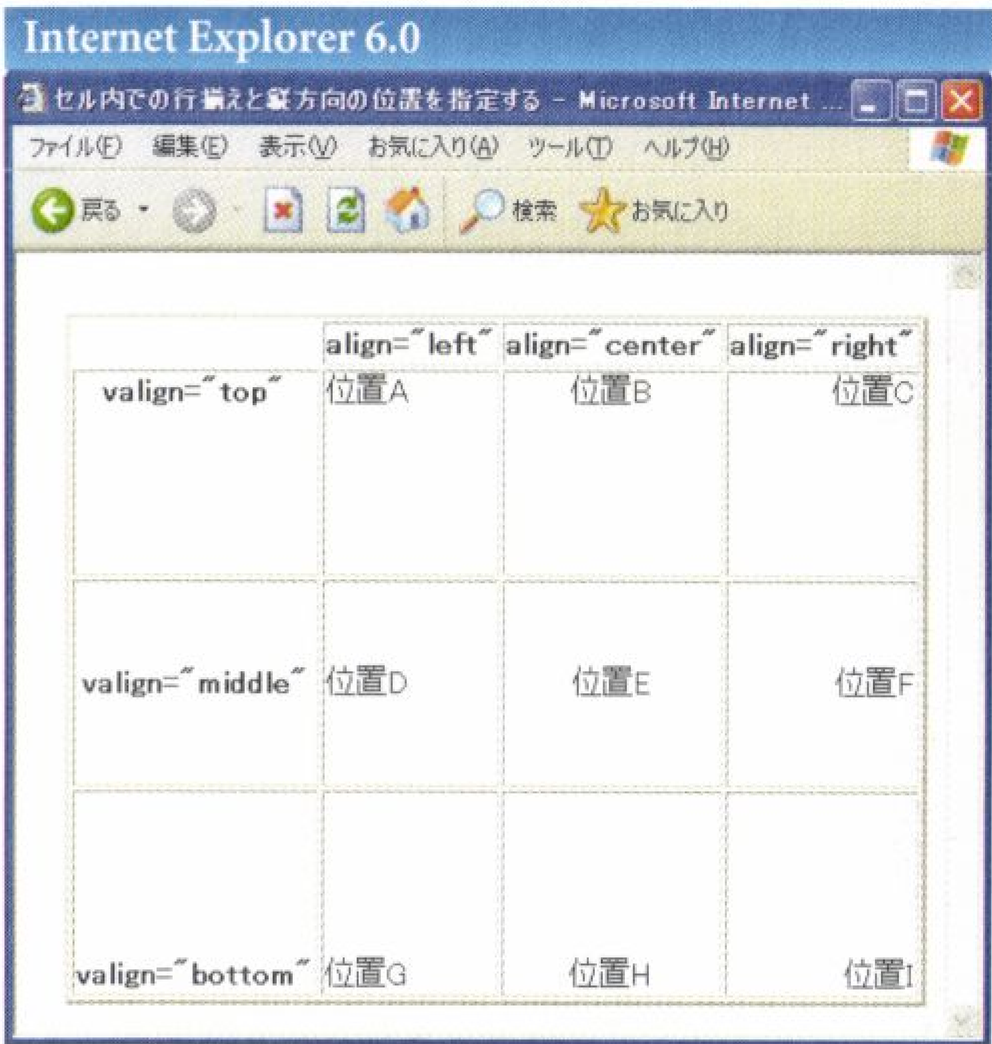

スタイルシート:「ボックス」の「内容の周りの空間を設定する」(P.239)

# セル内での行揃えと縦方向の位置を指定する

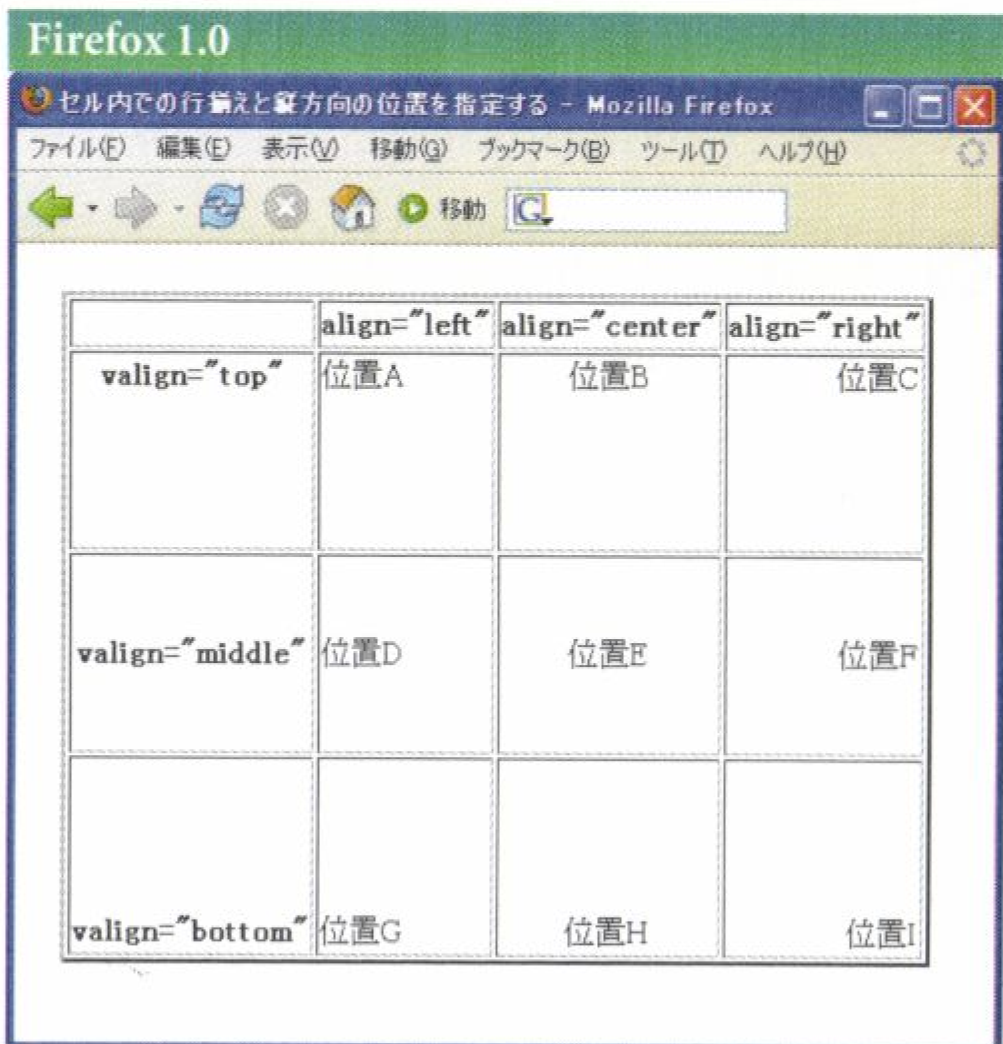
```
<tr align="行揃え位置" valign="縦位置">～</tr>
<th align="行揃え位置" valign="縦位置">～</th>
<td align="行揃え位置" valign="縦位置">～</td>
<thead align="行揃え位置" valign="縦位置">～</thead>
<tbody align="行揃え位置"valign="縦位置">～</tbody>
<tfoot align="行揃え位置" valign="縦位置">～</tfoot>
<col align="行揃え位置" valign="縦位置">～</col>
<colgroup align="行揃え位置"valign="縦位置">～</colgroup>
```

| 【行揃え位置】 | |
|---|---|
| left | 左揃え（td 要素のデフォルト） |
| right | 右揃え |
| center | 中央揃え（th 要素のデフォルト） |
| justify | 両端揃え |
| **【縦位置】** | |
| top | 上 |
| middle | 中央（デフォルト） |
| bottom | 下 |
| baseline | 横方向の列での 1 行目のベースライン |

Internet Explorer 6.0




Firefox 1.0




align 属性はセル内での行揃えを、valign 属性はセル内での縦方向の表示位置を設定します。
スタイルシートにも同様の効果のあるプロパティ（text-align と vertical-align）があります。

## Sample

```
<table border="2">
  <tr>
    <th></th>
    <th>align="left"</th>
    <th>align="center"</th>
    <th>align="right"</th>
  </tr>
  <tr valign="top">
    <th height="100">valign="top"</th>
    <td align="left">位置A</td>
    <td align="center">位置B</td>
    <td align="right">位置C</td>
  </tr>
  <tr valign="middle">
    <th height="100">valign="middle"</th>
    <td align="left">位置D</td>
    <td align="center">位置E</td>
    <td align="right">位置F</td>
  </tr>
  <tr valign="bottom">
    <th height="100">valign="bottom"</th>
    <td align="left">位置G</td>
    <td align="center">位置H</td>
    <td align="right">位置I</td>
  </tr>
</table>
```



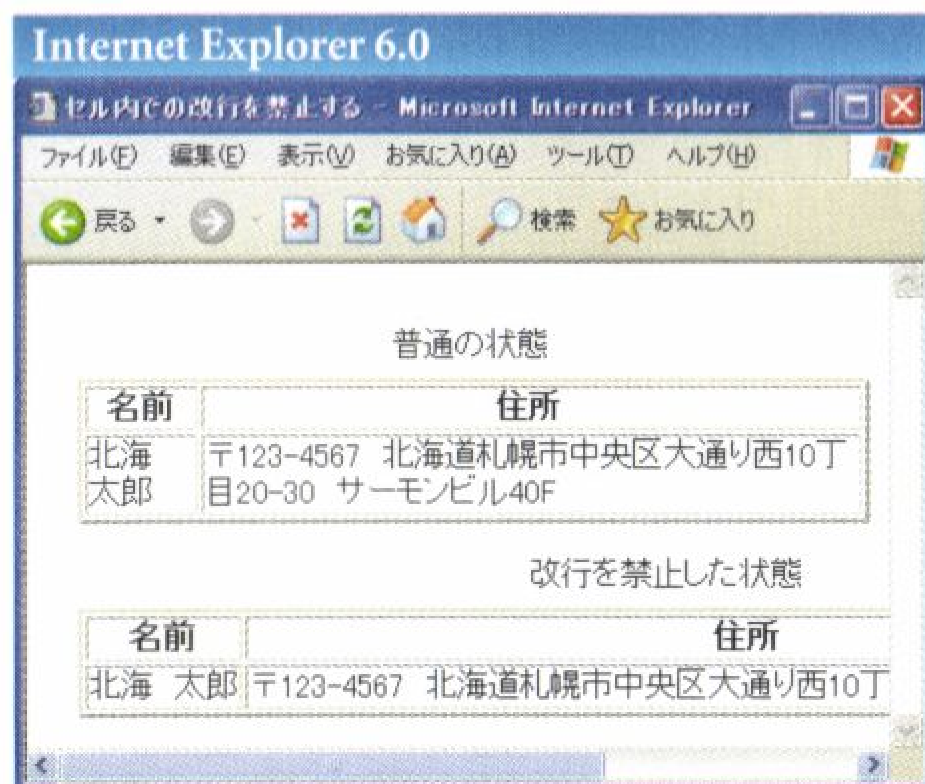
<thead>・<tbody>・<tfoot>：「テーブル」の「横列をグループ化する」(P.88)
<colgroup>：「テーブル」の「縦列をグループ化する」(P.90)
<col>：「テーブル」の「縦列に属性やスタイルシートを指定する」(P.92)
スタイルシート：「テキスト」の「行揃えを指定する」(P.214)
スタイルシート：「テキスト」の「縦方向の位置を指定する」(P.215)

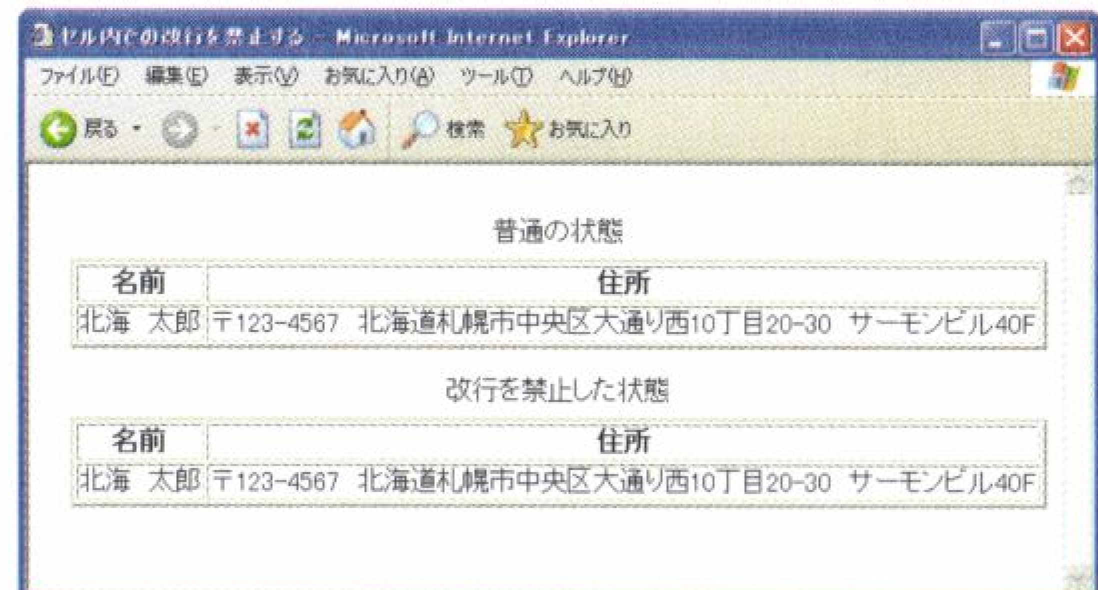
# セル内での改行を禁止する

**<th nowrap>～</th>**
**<td nowrap>～</td>**

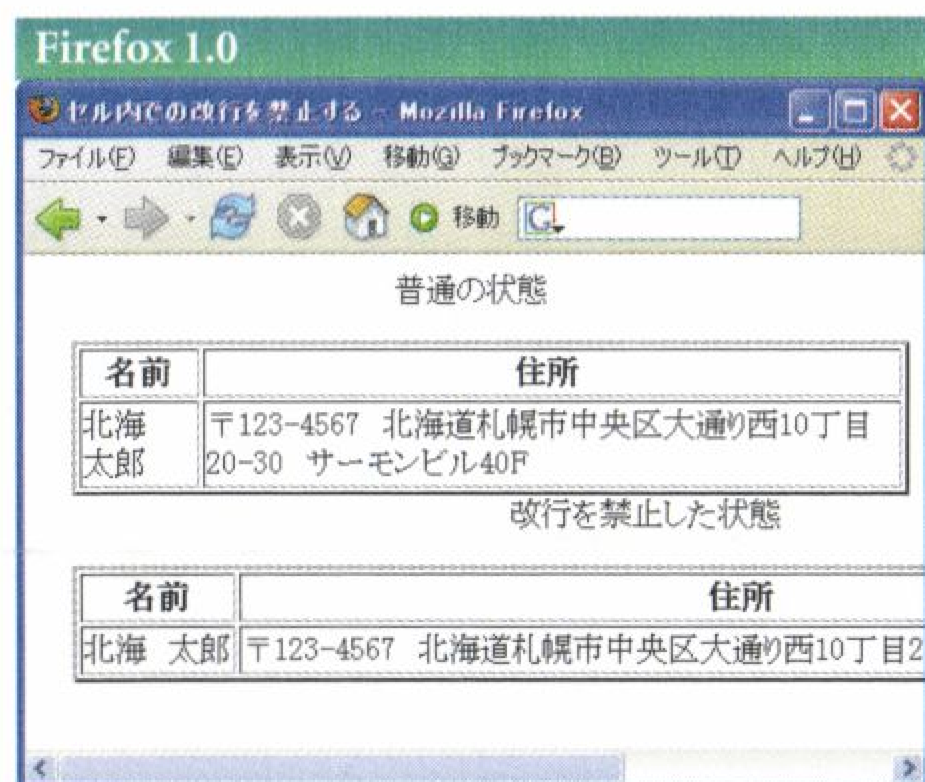


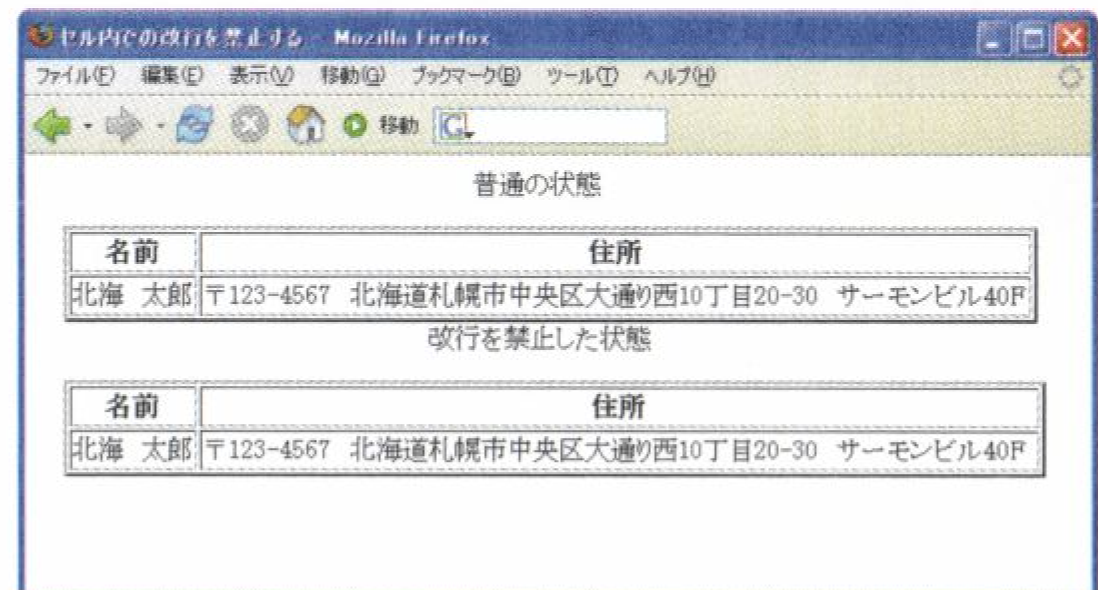
Internet Explorer 6.0




Firefox 1.0






通常、セルの表示幅はウィンドウの幅に合わせて自動的に調整されるため、セル内のテキストが長い場合には途中で改行されてしまうことがあります。そのような改行を禁止するのが、nowrap属性です。
ただし、nowrap属性を使用することは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。改行を禁止する場合には、できるだけスタイルシートを利用するようにしてください。

**Sample**

```
<table border="2">
<caption>普通の状態</caption>
<tr>
<th>名前</th>
<th>住所</th>
</tr>
<tr>
<td>北海　太郎</td>
<td>〒123-4567　北海道札幌市中央区大通り西10丁目20-30
```



```
サーモンビル40F</td>
</tr>
</table>

<table border="2">
<caption>改行を禁止した状態</caption>
<tr>
<th>名前</th>
<th>住所</th>
</tr>
<tr>
<td nowrap>北海　太郎</td>
<td nowrap>〒123-4567　北海道札幌市中央区大通り西10丁目20
```



```
-30　サーモンビル40F</td>
</tr>
</table>
```



スタイルシート：「テキスト」の「改行しないで表示させる」(P.221)

# 表やセルの背景色を指定する

```
<table bgcolor="色指定">～</table>
<tr bgcolor="色指定">～</tr>
<th bgcolor="色指定">～</th>
<td bgcolor="色指定">～</td>
```



Internet Explorer 6.0




Firefox 1.0




bgcolor属性を使用すると、背景色を指定することができます。
table要素に指定すると表全体に、tr要素に指定するとその横一列に、th要素とtd要素の場合は、そのセルに背景色が適用されます。
ただし、bgcolor属性を使用することは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。背景色を指定する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<table border="2" cellpadding="10">
  <tr bgcolor="#009933">
    <th>名前</th>
    <th>年齢</th>
    <th>性別</th>
  </tr>
  <tr>
    <td bgcolor="#ffcc00">北海　太郎</td>
    <td>32</td>
    <td>男</td>
  </tr>
  <tr>
    <td bgcolor="#ffcc00">北海　道子</td>
    <td>25</td>
    <td>女</td>
  </tr>
</table>
```

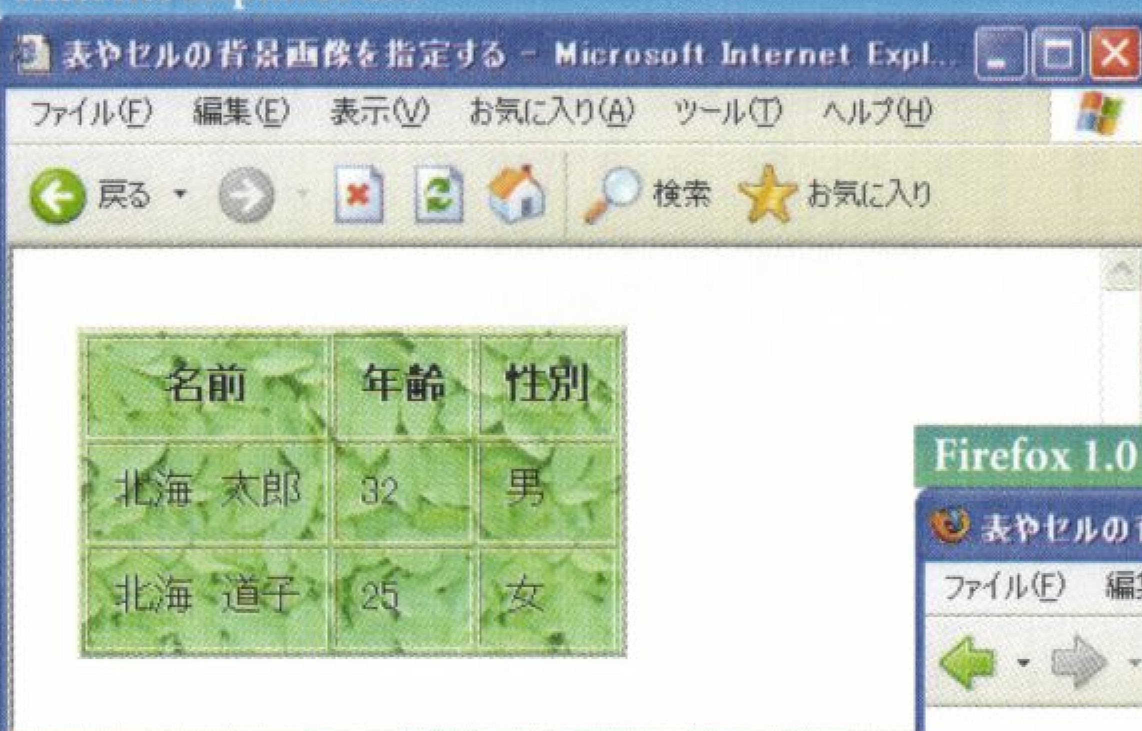

色指定:「HTMLについて」の「色の指定方法」(P.6)
スタイルシート:「背景」の「背景色を指定する」(P.224)
色指定:巻末付録「カラーチャート1～3」(巻末)

HTML

# 表やセルの背景画像を指定する

```
<table background="背景画像のURL">～</table>
<tr background="背景画像のURL">～</tr>
<th background="背景画像のURL">～</th>
<td background="背景画像のURL">～</td>
```

Internet Explorer 6.0




Firefox 1.0




background属性を使用すると、背景画像を指定することができます。
table要素に指定すると表全体に、tr要素に指定するとその横一列に、th要素とtd要素の場合はそのセルに背景画像が適用されます。
ただし、この属性は一部のブラウザでは利用できますが、標準的な仕様では定義されていないものです。背景画像を指定したい場合には、スタイルシートを利用してください。

## Sample

```
<table border="2" background="leaves.jpg" cellpadding="10">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>
```

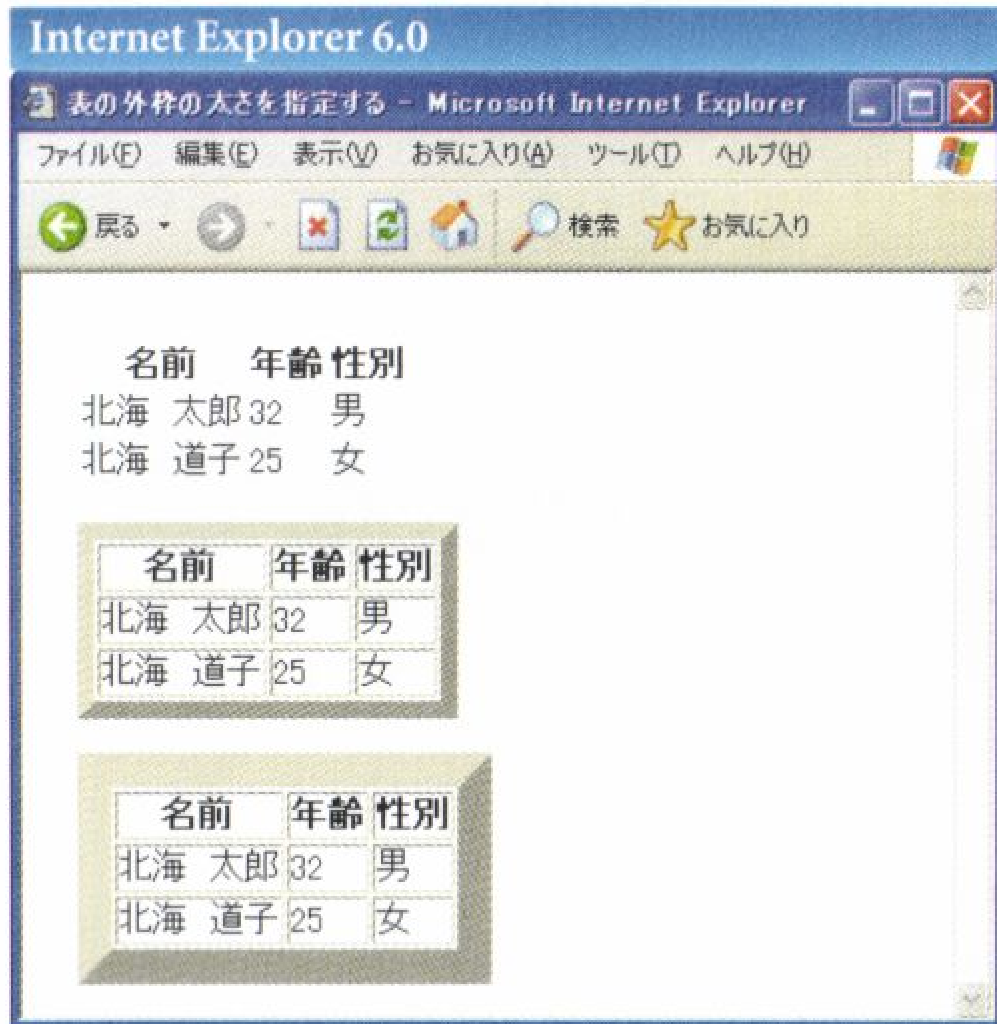

スタイルシート：「背景」の「背景画像を指定する」(P.227)

# 表の外枠の太さを指定する

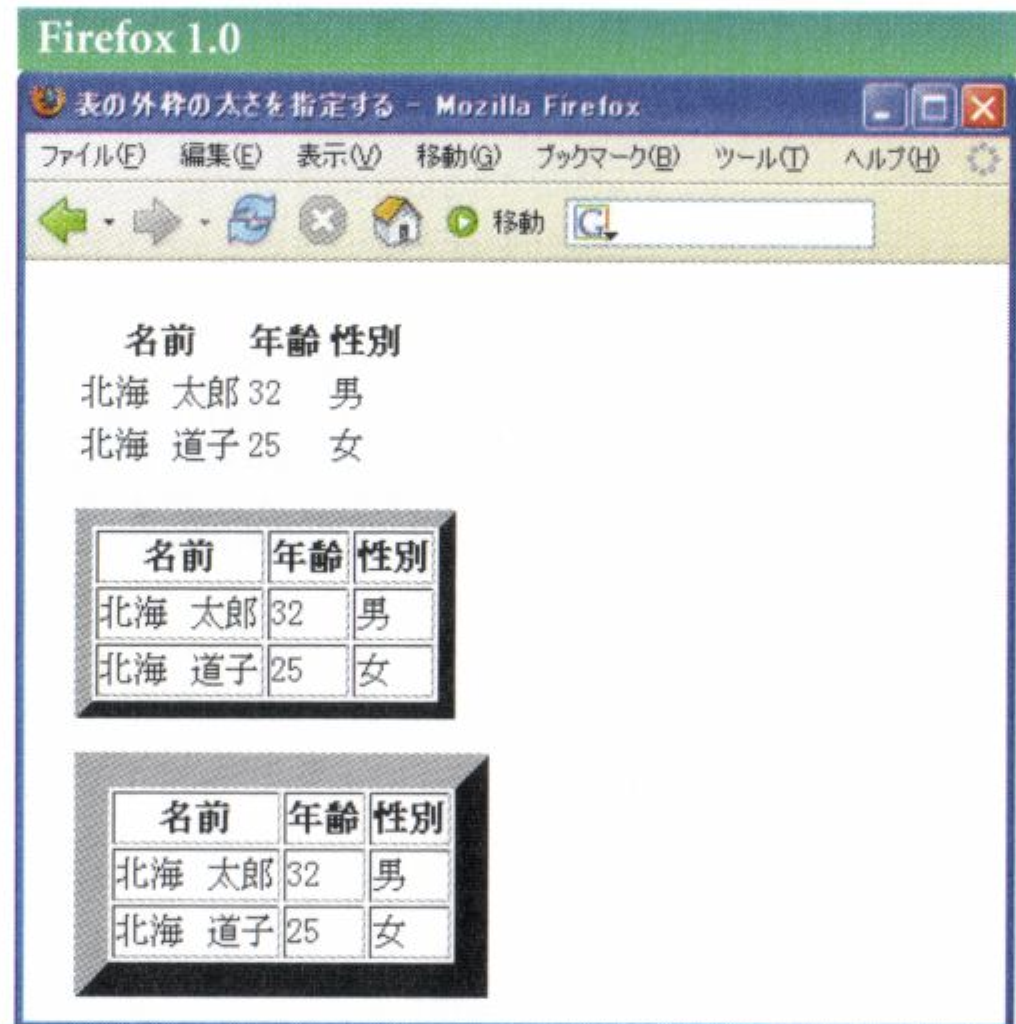
**<table border="外枠の太さ">～</table>**

外枠の太さ　ピクセル

Internet Explorer 6.0



Firefox 1.0



border属性は、表全体の外枠の太さを設定します。
値として0を指定すると、枠が表示されなくなります。スタイルシートにも同様の効果のあるプロパティ(border-widthなど)があります。

## Sample

```
<table border="0">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>

<table border="8">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>

<table border="16">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>
```


スタイルシート：「ボックス」の「枠線の太さを指定する」(P.241)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X Opera7 IE5-mac IE4-mac

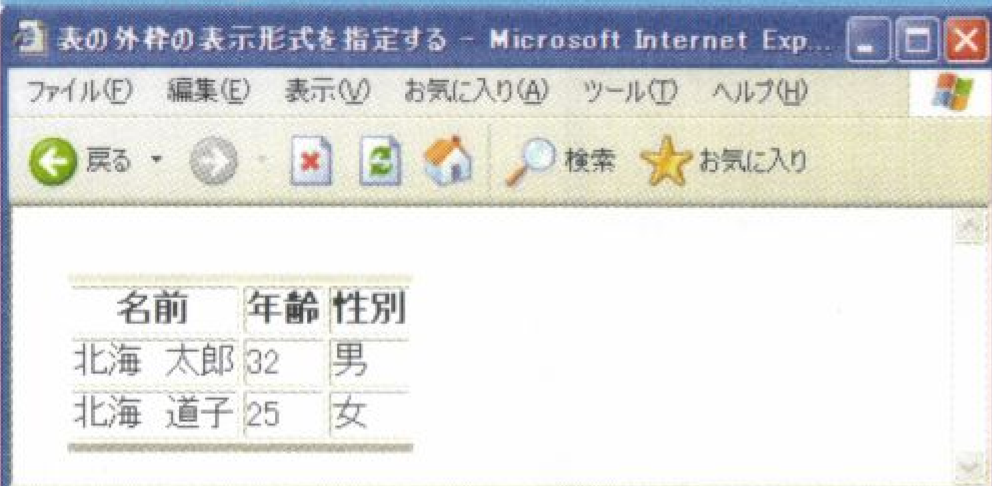
# 表の外枠の表示形式を指定する

## <table frame="外枠の表示形式">～</table>

【外枠の表示形式】

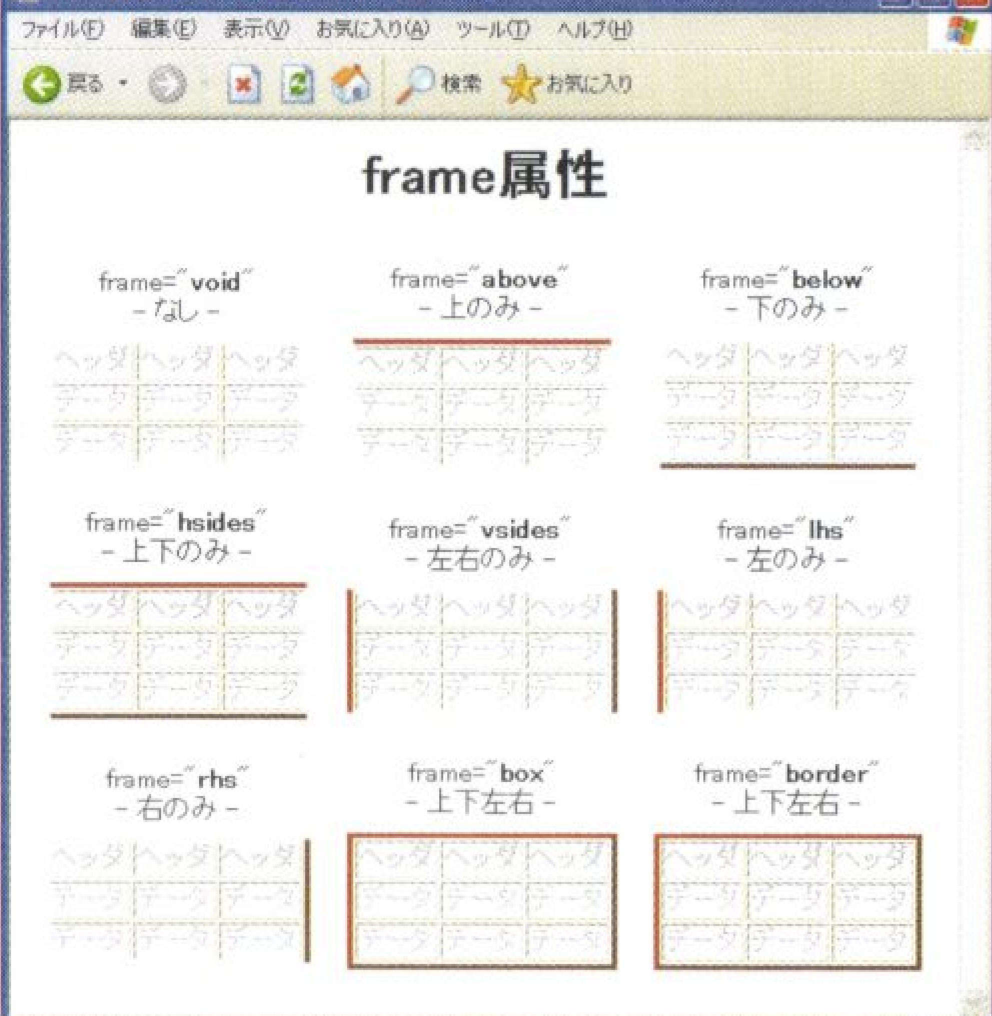
| | | | |
|---|---|---|---|
| void | なし(デフォルト) | above | 上のみ |
| below | 下のみ | lhs | 左のみ |
| rhs | 右のみ | hsides | 上下のみ |
| vsides | 左右のみ | box | 上下左右 |
| border | 上下左右 | | |

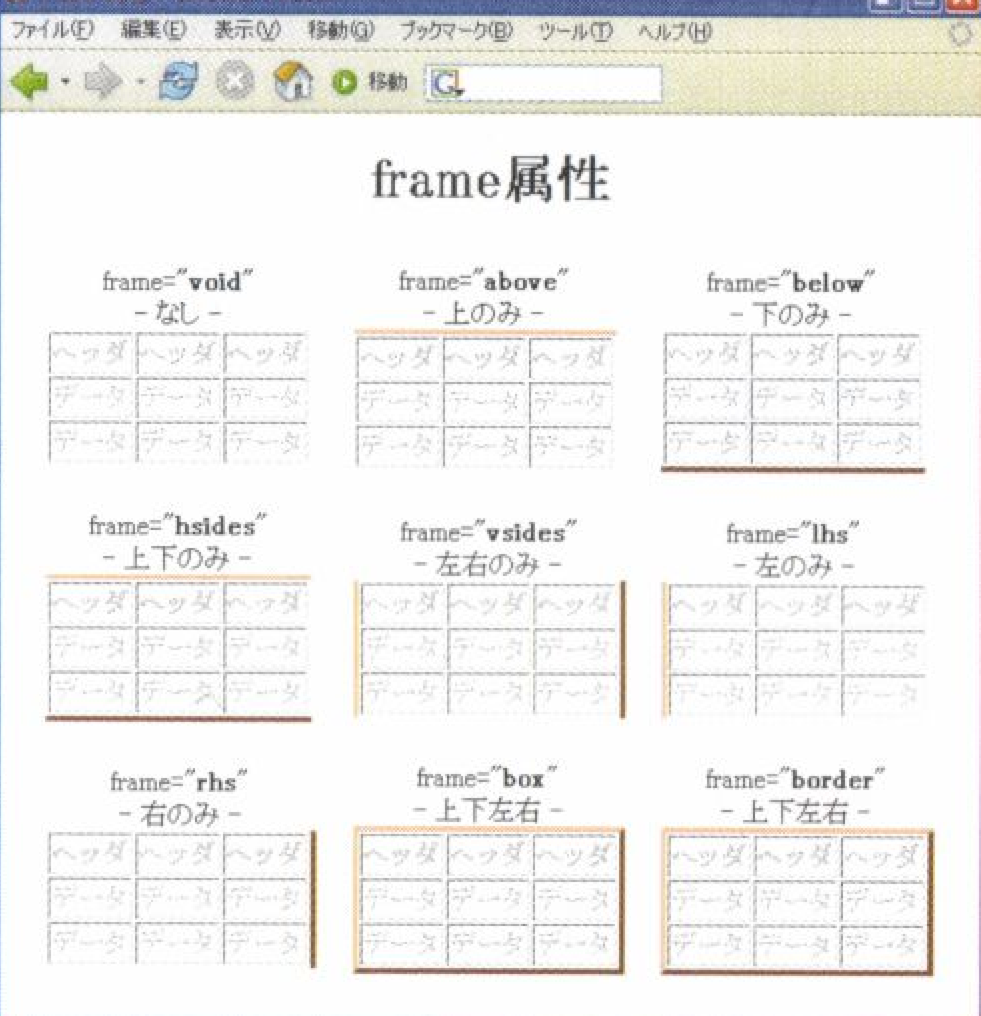
Internet Explorer 6.0




Firefox 1.0




frame属性は、表全体を囲う外枠のうち、上下左右のどの枠を表示させるかを設定します。

## Sample

```
<table border="4" frame="hsides">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>
```

表を作る

# 表内の枠線の表示形式を指定する

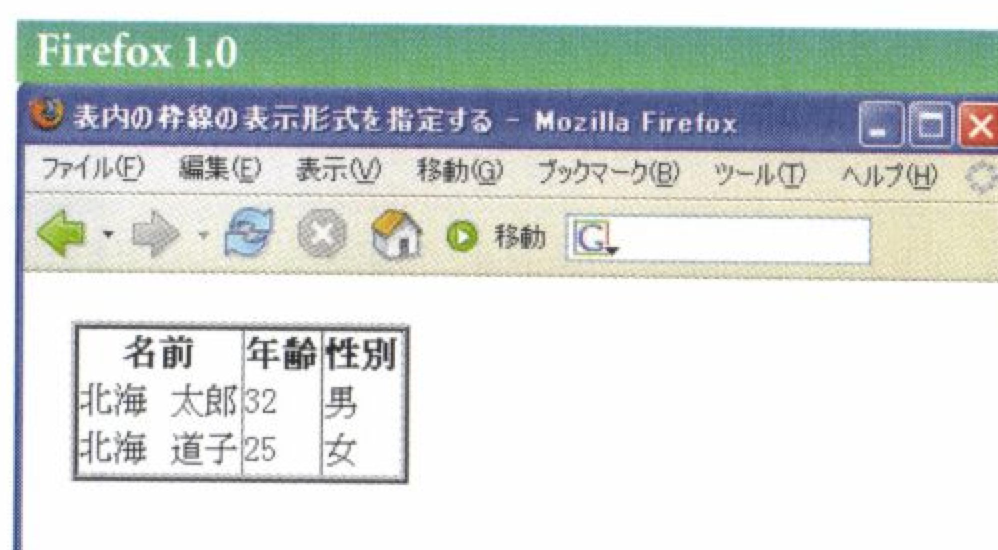
## <table rules="セルを区切る線の表示形式">～</table>

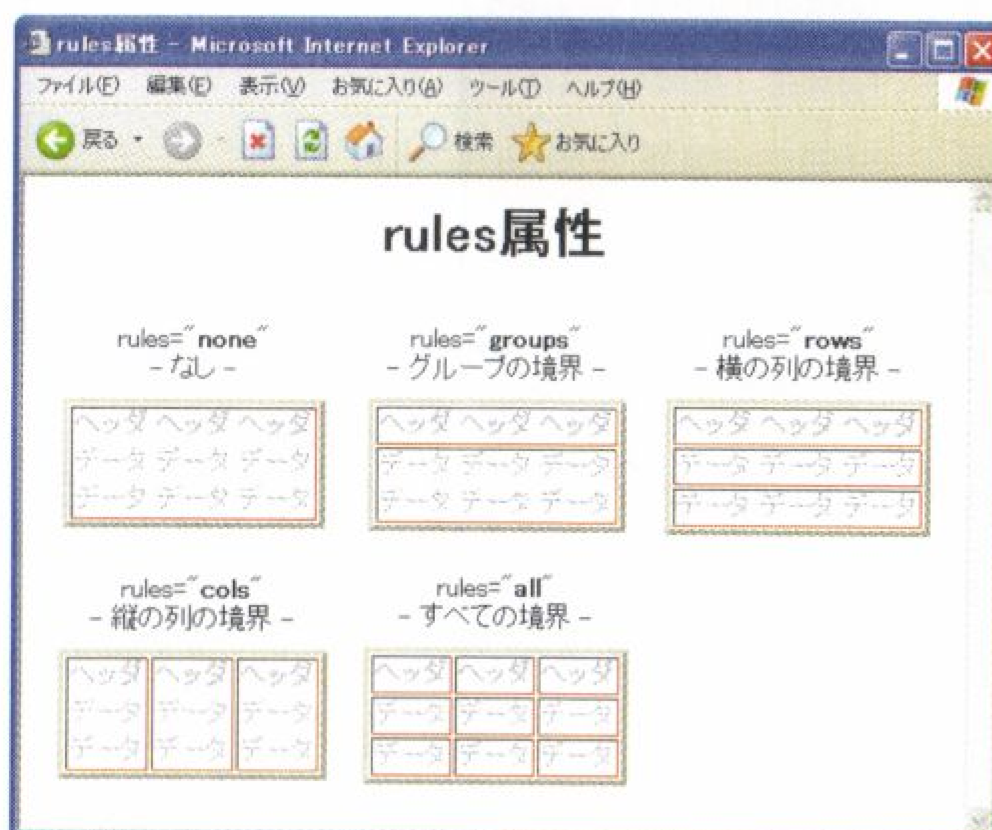
| 【セルを区切る線の表示形式】 | |
|---|---|
| none | なし(デフォルト) |
| rows | 横列の境界のみ |
| cols | 縦列の境界のみ |
| groups | thead要素・tbody要素・tfoot要素・colgroup要素・col要素の境界のみ |
| all | すべての境界 |

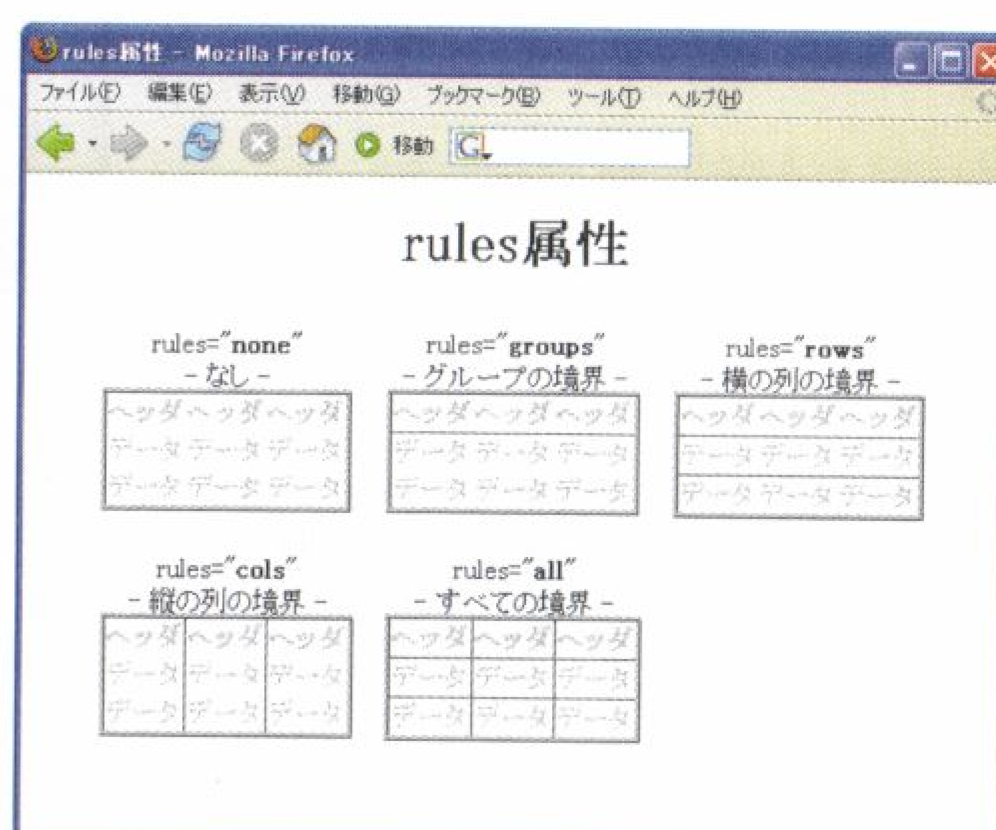
Internet Explorer 6.0




Firefox 1.0




rules属性は、セルを区切る線のうち、どの線を表示させるかを設定します。

### Sample

```
<table border="4" rules="cols">
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>
```


<thead>・<tbody>・<tfoot>：「テーブル」の「横列をグループ化する」(P.88)
<colgroup>：「テーブル」の「縦列をグループ化する」(P.90)
<col>：「テーブル」の「縦列に属性やスタイルシートを指定する」(P.92)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 横列をグループ化する

**<thead>～</thead>** ←表のヘッダ部分
**<tbody>～</tbody>** ←表の本体(データ)部分
**<tfoot>～</tfoot>** ←表のフッタ部分

Internet Explorer 6.0

横列をグループ化する - Microsoft Internet Explorer

| 集客数 | 札幌店 | 旭川店 | 小樽店 |
|---|---|---|---|
| 4月 | 2,085 | 1,605 | 1,124 |
| 5月 | 1,866 | 1,289 | 1,002 |
| 6月 | 1,924 | 1,455 | 1,191 |
| 合計 | 5,875 | 4,349 | 3,317 |

Firefox 1.0

横列をグループ化する - Mozilla Firefox

| 集客数 | 札幌店 | 旭川店 | 小樽店 |
|---|---|---|---|
| 4月 | 2,085 | 1,605 | 1,124 |
| 5月 | 1,866 | 1,289 | 1,002 |
| 6月 | 1,924 | 1,455 | 1,191 |
| 合計 | 5,875 | 4,349 | 3,317 |

thead要素・tbody要素・tfoot要素は、いずれも表の横列(tr要素)をグループ化する要素です。グループ化する部分が表のヘッダであればthead要素を、表の本体部分であればtbody要素を、表のフッタであればtfoot要素を使用します。
このようにグループ化しておくことで、そのグループに対して属性やスタイルシートをまとめて適用できるようになります。また、ブラウザの対応次第では、ヘッダとフッタを固定した状態で本体部分をスクロールさせることや、複数ページに渡る長い表を印刷する場合に各ページにヘッダとフッタを印刷させることなども可能になります。
これらの要素を使用する場合は、必ずthead要素、tfoot要素、tbody要素の順になるようにしてください。tbody要素よりも前にtfoot要素を配置するのは、どの程度の長さかわからない本体よりも先にフッタを表示できるようにするためです。ただし、tfoot要素に未対応のブラウザでは、フッタをそのまま本体の前に表示してしまいますので、注意してください。なお、thead要素とtfoot要素は、ひとつのテーブル内にひとつずつしか配置できませんが、tbody要素は必要に応じて複数配置することができます。

**Sample**

```
<table border="3" cellpadding="4">
<thead>
  <tr>
  <th>集客数</th><th>札幌店</th><th>旭川店</th><th>小樽店</th>
  </tr>
</thead>
<tfoot>
  <tr>
  <th>合計</th><td>5,875</td><td>4,349</td><td>3,317</td>
  </tr>
</tfoot>
<tbody>
  <tr>
  <td>4月</td><td>2,085</td><td>1,605</td><td>1,124</td>
  </tr>
  <tr>
  <td>5月</td><td>1,866</td><td>1,289</td><td>1,002</td>
  </tr>
  <tr>
  <td>6月</td><td>1,924</td><td>1,455</td><td>1,191</td>
  </tr>
</tbody>
</table>
```

**【スタイルシート】**

```
td { text-align: right }
thead, tfoot th {
  color: #ffffff;
  background-color: #009933
}
tbody {
  color: #000000;
  background-color: #ffcc00
}
```


TIPS「表のヘッダとフッタを固定して内容をスクロールさせる」(P.91)

# 縦列をグループ化する

<colgroup span="*縦列数*">～</colgroup>
<colgroup span="*縦列数*" width="*幅*">～</colgroup>

| | |
|---|---|
| 縦列数 | グループ化する縦列の数(省略した場合は 1) |
| 幅 | ピクセル･%･* |

**Internet Explorer 6.0**

縦列をグループ化する - Microsoft Internet Explorer

2005年の釣果(匹)

| 日付 | 場所 | アイナメ | ヒラメ | ソイ |
|---|---|---|---|---|
| 6月01日 | 余市港 | 0 | 0 | 12 |
| 6月15日 | 小樽港 | 2 | 0 | 8 |
| 6月26日 | 美国漁港 | 4 | 2 | 23 |
| 7月04日 | 小樽港 | 0 | 3 | 5 |

**Firefox 1.0**

縦列をグループ化する - Mozilla Firefox

2005年の釣果(匹)

| 日付 | 場所 | アイナメ | ヒラメ | ソイ |
|---|---|---|---|---|
| 6月01日 | 余市港 | 0 | 0 | 12 |
| 6月15日 | 小樽港 | 2 | 0 | 8 |
| 6月26日 | 美国漁港 | 4 | 2 | 23 |
| 7月04日 | 小樽港 | 0 | 3 | 5 |

span属性は、表の縦列を(構造的な意味で)グループ化する場合に利用します。これによって、複数の縦列に対して、幅や行揃えなどの属性やスタイルシートをまとめて指定できるようになります。
設定したグループ内で、さらに縦列に対して個別の指定をしたい場合には、次項で説明するcol要素を使用します。その場合は、colgroup要素でspan属性を指定せずに、col要素側で指定するようにしてください。colgroup要素は、配置する位置に注意が必要です。caption要素の直後(なければtable要素の開始タグの直後)で、thead要素やtr要素よりも前の位置に配置するようにしてください。また、この要素の内容として配置できるのは、col要素だけです。

## Sample

```
<table border="3">
<caption>2005年の釣果(匹)</caption>
<colgroup span="2" id="dateplace"></colgroup>
<colgroup span="3" id="count" width="100" align="right"></colgroup>
<tr>
<th>日付</th><th>場所</th><th>アイナメ</th><th>ヒラメ</th><th>ソイ</th>
</tr>
<tr>
<td>6月01日</td><td>余市港</td><td>0</td><td>0</td><td>12</td>
</tr>
<tr>
<td>6月15日</td><td>小樽港</td><td>2</td><td>0</td><td>8</td>
</tr>
<tr>
<td>6月26日</td><td>美国漁港</td><td>4</td><td>2</td><td>23</td>
```

```
</tr>
<tr>
<td>7月04日</td><td>小樽港</td><td>0</td><td>3</td><td>5</td>
</tr>
</table>
```

**【スタイルシート】**

```
colgroup#dateplace {
  color: #000000;
  background-color: #ffcc00
}
th { text-align: center }
```

TIPS「『*』による幅や高さの割合の指定」(P.137)

## TIPS

### 表のヘッダとフッタを固定して内容をスクロールさせる

| | 札幌 | 小樽 | 旭川 |
|---|---|---|---|
| 4月 | 914 | 212 | 2[illegible] |
| 5月 | 1,024 | 198 | 3[illegible] |
| 合計 | 8,652 | 1,654 | 2,940 |

Netscape 6.0以降では、スタイルシートを利用することで、表のヘッダとフッタを固定した状態で本体をスクロールさせることができます。スタイルシートの指定は、以下のようにtbody要素のoverflowプロパティの値を「auto」に設定して、あとはheightプロパティで高さを指定すればOKです。

```
tbody { overflow: auto; height: 100px }
```

ただし、このように指定すると、Windows版のInternet Explorerでは各tr要素の高さがtbody要素に指定した高さになってしまいます。これを回避するためには、同時に次のような指定も入れておいてください。

```
tr { height: auto }
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X Opera7 IE5-mac

# 縦列に属性やスタイルシートを指定する

<col span="*縦列数*">
<col span="*縦列数*" width="*幅*">

| | |
|---|---|
| 縦列数 | 属性をまとめて指定する縦列の数(省略した場合は1) |
| 幅 | ピクセル·%·* |

Internet Explorer 6.0

縦列に属性やスタイルシートを指定する - Microsoft Internet Explorer

2005年の釣果(匹)

| 日付 | 場所 | アイナメ | ヒラメ | ソイ |
|---|---|---|---|---|
| 6月01日 | 余市港 | 0 | 0 | 12 |
| 6月15日 | 小樽港 | 2 | 0 | 8 |
| 6月26日 | 美国漁港 | 4 | 2 | 23 |
| 7月04日 | 小樽港 | 0 | 3 | 5 |

Firefox 1.0

縦列に属性やスタイルシートを指定する - Mozilla Firefox

2005年の釣果(匹)

| 日付 | 場所 | アイナメ | ヒラメ | ソイ |
|---|---|---|---|---|
| 6月01日 | 余市港 | 0 | 0 | 12 |
| 6月15日 | 小樽港 | 2 | 0 | 8 |
| 6月26日 | 美国漁港 | 4 | 2 | 23 |
| 7月04日 | 小樽港 | 0 | 3 | 5 |



col要素は、表の縦列を構造的にグループ化するのではなく、縦列に対して幅や行揃えなどの属性やスタイルシートをまとめて指定したい場合に利用します。
配置する位置は、caption要素の直後(なければtable要素の開始タグの直後)で、thead要素やtr要素よりも前になるようにしてください。colgroup要素がある場合は、その内容として配置することもできます。

## Sample

```
<table border="3">
<caption>2005年の釣果(匹)</caption>
<col id="date">
<col id="place">
<col span="3" id="count" width="100" align="right">
<tr>
<th>日付</th><th>場所</th><th>アイナメ</th><th>ヒラメ</th><th>ソイ</th>
</tr>
<tr>
<td>6月01日</td><td>余市港</td><td>0</td><td>0</td><td>12</td>
</tr>
<tr>
<td>6月15日</td><td>小樽港</td><td>2</td><td>0</td><td>8</td>
</tr>
<tr>
<td>6月26日</td><td>美国漁港</td><td>4</td><td>2</td><td>23</td>
</tr>
<tr>
<td>7月04日</td><td>小樽港</td><td>0</td><td>3</td><td>5</td>
</tr>
</table>
```

**【スタイルシート】**

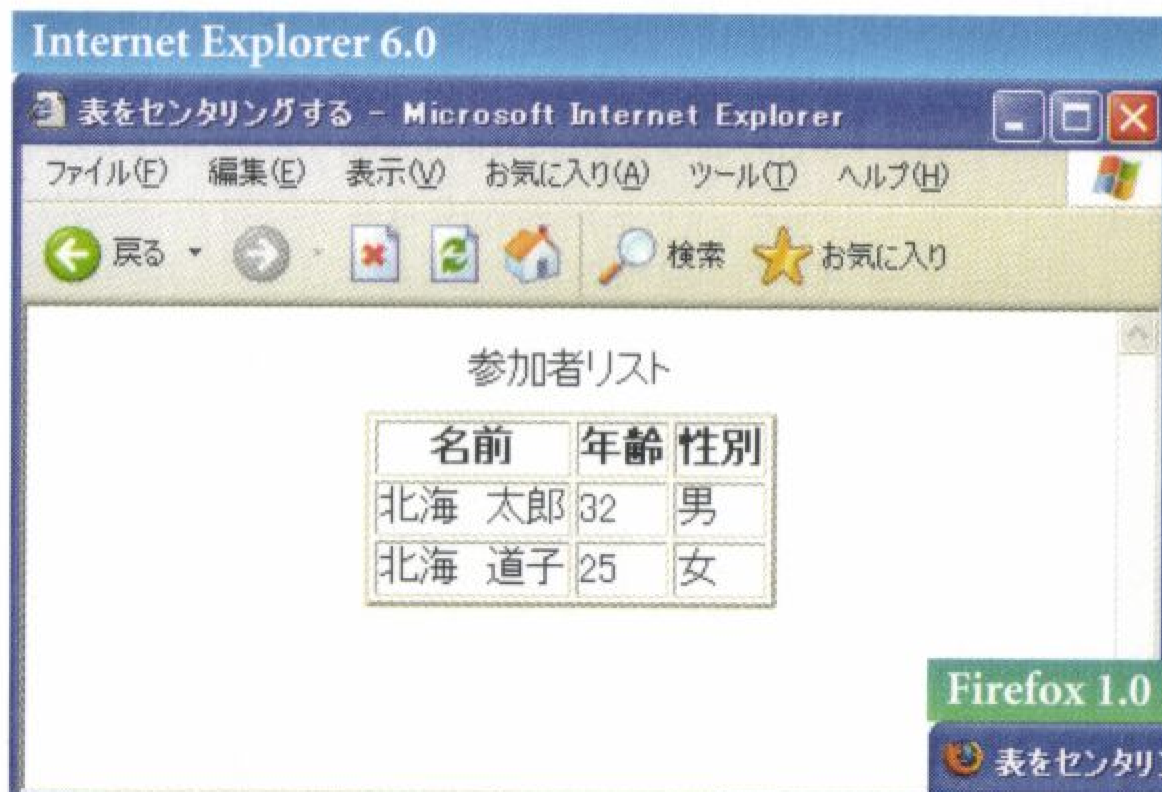
```
col#date {
  color: #000000;
  background-color: #009933
}
col#place {
  color: #000000;
  background-color: #ffcc00
}
th { text-align: center }
```


TIPS「『*』による幅や高さの割合の指定」(P.137)

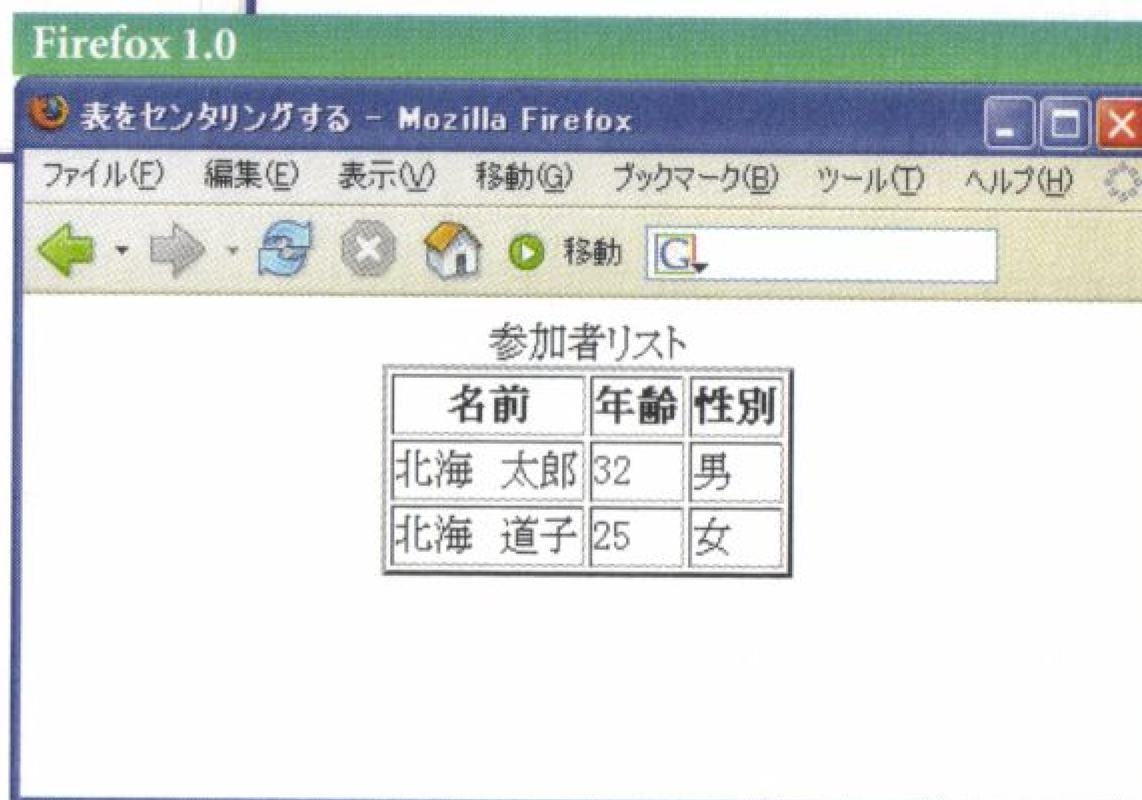
# 表をセンタリングする

## <table align="center">～</table>

Internet Explorer 6.0




Firefox 1.0




table要素のalign属性でcenterを指定すると、表をセンタリングして表示します。表をcenter要素で囲った場合と同様の結果になります。
ただし、align属性を使用することは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。センタリングしたい場合には、できるだけスタイルシートを利用するようにしてください。

### Sample

```
<table border="2" align="center">
<caption>参加者リスト</caption>
<tr><th>名前</th><th>年齢</th><th>性別</th></tr>
<tr><td>北海　太郎</td><td>32</td><td>男</td></tr>
<tr><td>北海　道子</td><td>25</td><td>女</td></tr>
</table>
```

<center>：「スタイルとレイアウト」の「センタリングする」(P.47)
スタイルシート：「表示と配置」の「センタリングする」(P.256)

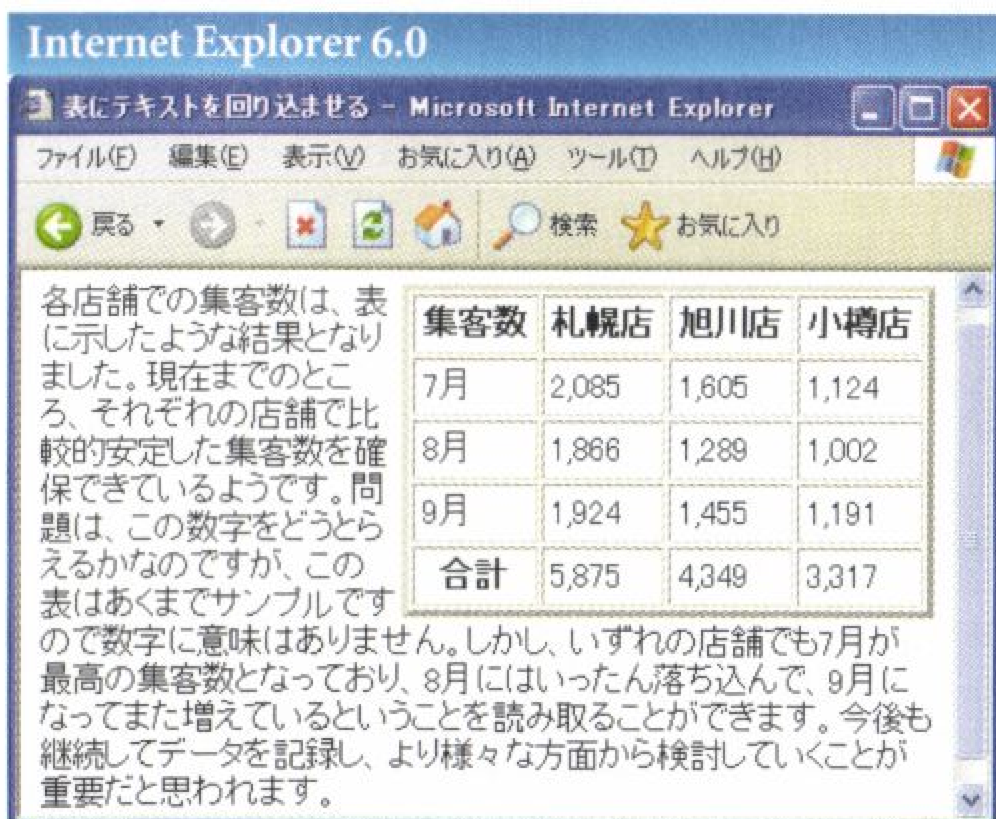
# 表にテキストを回り込ませる

## <table align="位置">～</table>

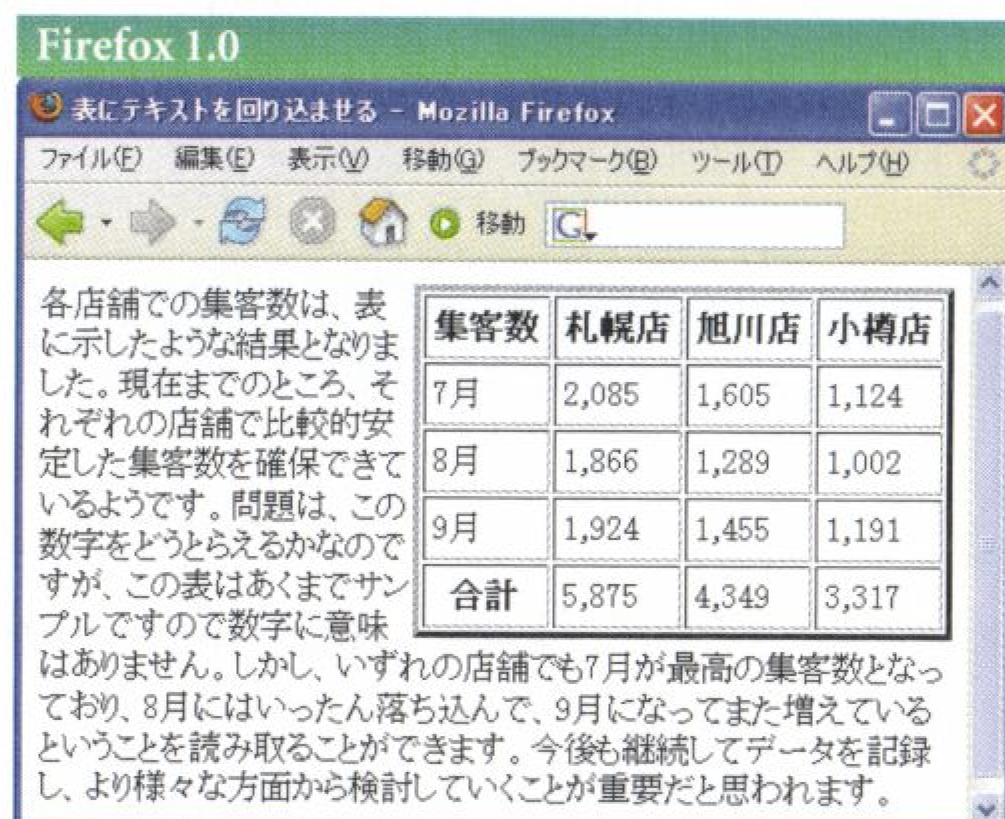
IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

| 位置 | left・right |
|---|---|

Internet Explorer 6.0




Firefox 1.0




table要素のalign属性でleftかrightを指定すると、表を左または右に配置して、その横にテキストを回り込ませます。
回り込みを解除したい場合には、<br>のclear属性を使用してください。
ただし、align属性を使うことは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。テキストを回り込ませる場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<table border="3" cellpadding="4" align="right">
<thead>
<tr><th>集客数</th><th>札幌店</th><th>旭川店</th><th>小樽店</th></tr>
</thead>
<tfoot>
<tr><th>合計</th><td>5,875</td><td>4,349</td><td>3,317</td></tr>
</tfoot>
<tbody>
<tr><td>7月</td><td>2,085</td><td>1,605</td><td>1,124</td></tr>
<tr><td>8月</td><td>1,866</td><td>1,289</td><td>1,002</td></tr>
<tr><td>9月</td><td>1,924</td><td>1,455</td><td>1,191</td></tr>
</tbody>
</table>
<p>
各店舗での集客数は、表に示したような結果となりました。(中略)重要だと思われます。
<br clear="right">
</p>
```

 スタイルシート:「表示と配置」の「左右への配置と回り込みを指定する」(P.252)

表を作る

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

# 表への回り込みを解除する

**<br clear="どちら側の表に対して解除するか">**

| 【どちら側の表に対して解除するか】 | |
|---|---|
| left | 左側の表に対する回り込みを解除 |
| right | 右側の表に対する回り込みを解除 |
| all | 両側の表に対する回り込みを解除 |

Internet Explorer 6.0

表への回り込みを解除する - Microsoft Internet Explorer

各店舗での集客数は、表に示したような結果となりました。現在までのところ、それぞれの店舗で比較的安定した集客数を確保できているようです。

| 集客数 | 札幌店 | 旭川店 | 小樽店 |
|---|---|---|---|
| 7月 | 2,085 | 1,605 | 1,124 |
| 8月 | 1,866 | 1,289 | 1,002 |
| 9月 | 1,924 | 1,455 | 1,191 |
| 合計 | 5,875 | 4,349 | 3,317 |

問題は、この数字をどうとらえるかなのですが、この表はあくまでサンプルですので数字に意味はありません。しかし、いずれの店舗でも7月が最高の集客数となっており、8月にはいったん落ち込んで、9月になってまた増えているということを読み取ることができます。今後も継続してデータを記録し、より様々な方面から検討していくことが重要だと思われます。

Firefox 1.0

表への回り込みを解除する - Mozilla Firefox

各店舗での集客数は、表に示したような結果となりました。現在までのところ、それぞれの店舗で比較的安定した集客数を確保できているようです。

| 集客数 | 札幌店 | 旭川店 | 小樽店 |
|---|---|---|---|
| 7月 | 2,085 | 1,605 | 1,124 |
| 8月 | 1,866 | 1,289 | 1,002 |
| 9月 | 1,924 | 1,455 | 1,191 |
| 合計 | 5,875 | 4,349 | 3,317 |

問題は、この数字をどうとらえるかなのですが、この表はあくまでサンプルですので数字に意味はありません。しかし、いずれの店舗でも7月が最高の集客数となっており、8月にはいったん落ち込んで、9月になってまた増えているということを読み取ることができます。今後も継続してデータを記録し、より様々な方面から検討していくことが重要だと思われます。

br要素のclear属性は、表の横にテキストを回り込ませている状態を解除します。この指定以降のテキストは、表の下の行から表示されるようになります。
ただし、clear属性を使うことは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。テキストの回り込みを解除する場合には、できるだけスタイルシートを利用するようにしてください。

表を作る

**Sample**

```
<table border="3" cellpadding="4" align="right">
<thead>
<tr><th>集客数</th><th>札幌店</th><th>旭川店</th><th>小樽店</th></tr>
</thead>
<tfoot>
<tr><th>合計</th><td>5,875</td><td>4,349</td><td>3,317</td></tr>
</tfoot>
<tbody>
<tr><td>7月</td><td>2,085</td><td>1,605</td><td>1,124</td></tr>
<tr><td>8月</td><td>1,866</td><td>1,289</td><td>1,002</td></tr>
<tr><td>9月</td><td>1,924</td><td>1,455</td><td>1,191</td></tr>
</tbody>
</table>
<p>
各店舗での集客数は、表に示したような結果となりました。現在までのところ、それぞれの店舗で比較的安定した集客数を確保できているようです。
<br clear="right">
問題は、この数字をどうとらえるかなのですが、この表はあくまでサンプルですので数字に意味はありません。しかし、いずれの店舗でも7月が最高の集客数となっており、8月にはいったん落ち込んで、9月になってまた増えているということを読み取ることができます。今後も継続してデータを記録し、よりさまざまな方面から検討していくことが重要だと思われます。
</p>
```

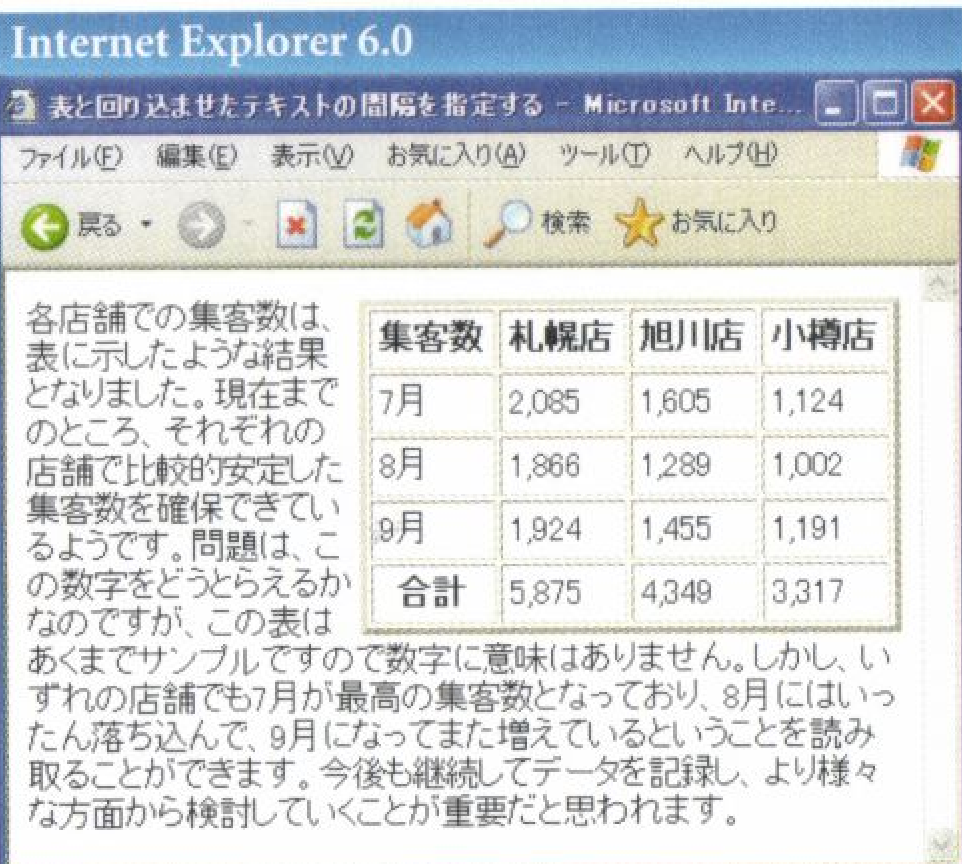

スタイルシート：「表示と配置」の「回り込みを解除する」(P.254)

# 表と回り込ませたテキストの間隔を指定する

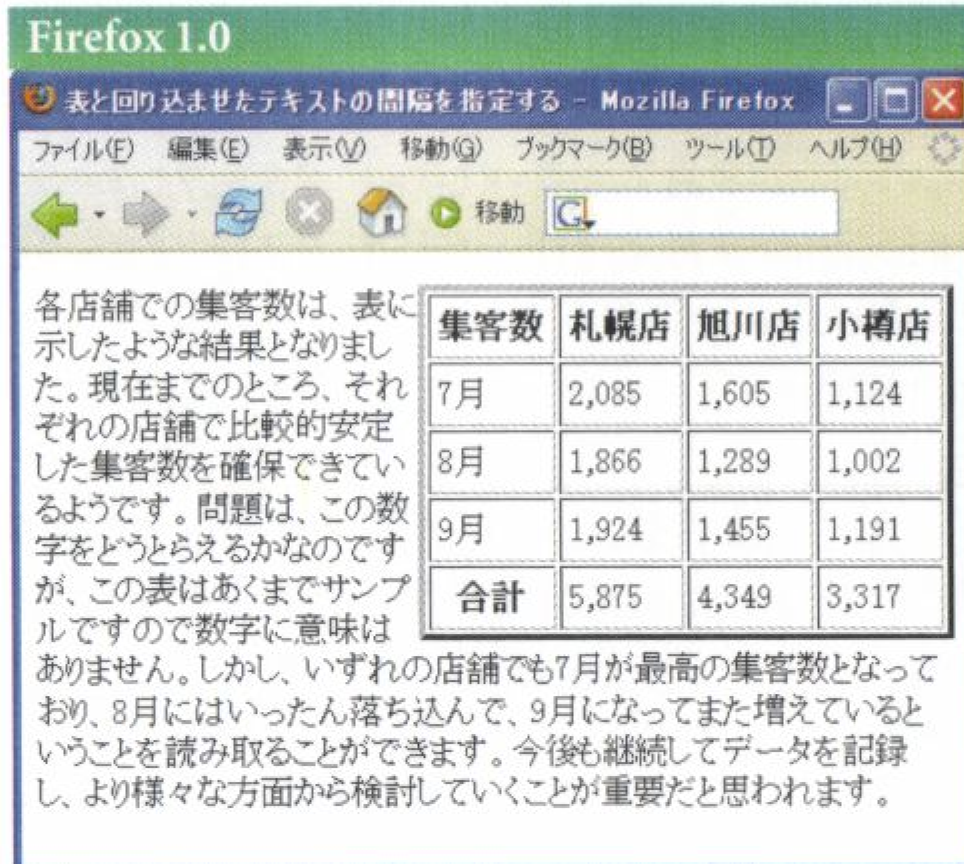
## <table vspace="*縦間隔*" hspace="*横間隔*">～</table>

| | |
|---|---|
| 縦間隔 | 表の上下の間隔(ピクセル) |
| 横間隔 | 表の左右の間隔(ピクセル) |

Internet Explorer 6.0




Firefox 1.0




表の横にテキストを回り込ませた場合の、表とテキストとの間隔を設定します。vspace属性を指定すると表の上下に、hspace属性を指定すると表の左右に指定したピクセル数の空間ができます。

ただし、これらの属性は一部のブラウザでは利用できますが、新しいHTMLの標準的な仕様では定義されていないものです。間隔を空けたい場合には、スタイルシートのマージンなどで調節するようにしてください。

### Sample

```
<table border="3" cellpadding="4" align="right" vspace="12" hspace="12">
<thead>
<tr><th>集客数</th><th>札幌店</th><th>旭川店</th><th>小樽店</th></tr>
</thead>
<tfoot>
<tr><th>合計</th><td>5,875</td><td>4,349</td><td>3,317</td></tr>
</tfoot>
<tbody>
<tr><td>7月</td><td>2,085</td><td>1,605</td><td>1,124</td></tr>
<tr><td>8月</td><td>1,866</td><td>1,289</td><td>1,002</td></tr>
<tr><td>9月</td><td>1,924</td><td>1,455</td><td>1,191</td></tr>
</tbody>
</table>
<p>
各店舗での集客数は、表に示したような結果となりました。(中略)重要だと思われます。
<br clear="right">
</p>
```

スタイルシート:「ボックス」の「マージンを設定する」(P.237)

N4.X

表を作る

# 画像を配置する

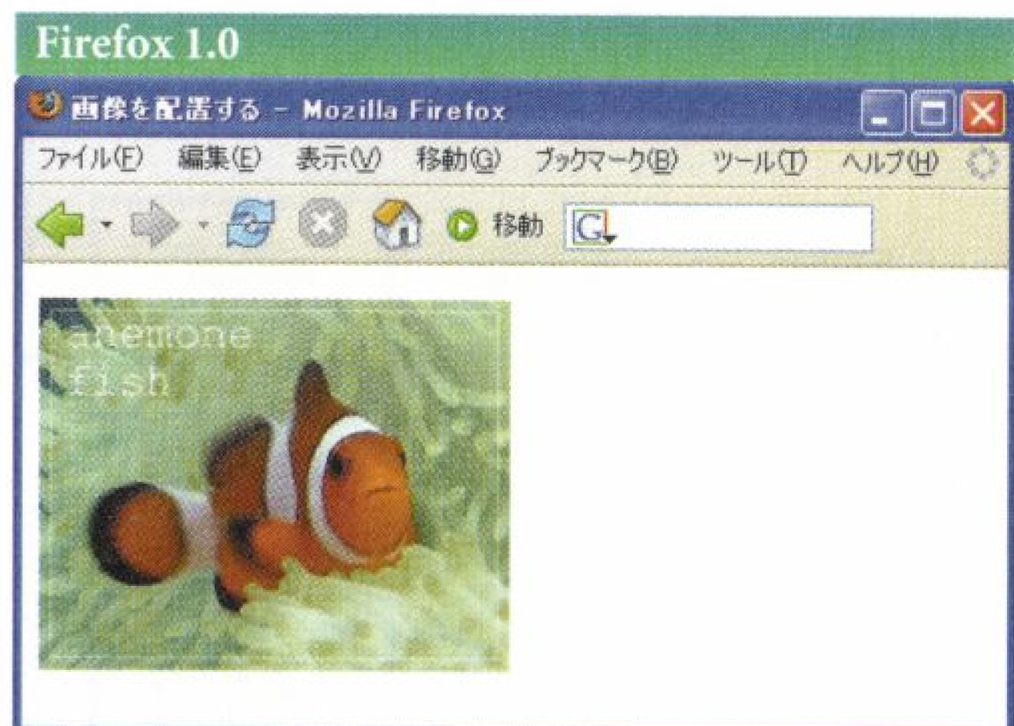
## <img src="*URL*" width="*幅*" height="*高さ*" alt="*代替テキスト*">

| | |
|---|---|
| URL | 画像ファイルのURL |
| 幅 | 画像の幅(ピクセル·%) |
| 高さ | 画像の高さ(ピクセル·%) |
| 代替テキスト | 画像の代わりの役目をするテキスト |

Internet Explorer 6.0




Firefox 1.0




img要素で、HTML文書内に画像を配置します。
画像の形式としては、一般にGIF形式·JPEG形式·PNG形式が利用できます。
幅と高さについては、画像の実際のサイズに関わらず、ここで指定した幅と高さで表示されます。これらの指定は必須ではありませんが、指定しておくと画像以降の内容を早く表示させることができます。
alt属性には、画像を表示できない環境で画像の代わりとして利用するテキストを指定します。内容としては、どのような画像なのかを説明するのではなく、その画像が表示されない場合に画像の代わりとしての役割を果たすような内容を、前後関係もよく考えた上で入れるようにしてください。なお、alt属性は省略できませんので、画像が特に意味のない飾りであるような場合には、内容を何も入れずに「alt=""」と指定してください。

### Sample

```
<p>
<img src="fish.jpg" width="220" height="170" alt="カクレクマノミ">
</p>
```

コラム「alt属性には適切な代替テキストを！」(P.108)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 画像の外枠を設定する

**<img src="*URL*" alt="*代替テキスト*" border="*枠の太さ*">**

| | |
|---|---|
| 枠の太さ | ピクセル |







img要素のborder属性は、画像の周りに表示される枠の太さを設定します。
一般的なブラウザでは、通常は枠が表示されていませんが、この属性を指定することで画像の周囲に枠が表示されるようになります。また、画像をリンクさせると、それを示すための枠が自動的に表示されますが、その場合の枠の太さも同様に設定できます。
ただし、border属性を使うことは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。枠の太さを設定する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<h1>リンクしている場合</h1>
<p>
※左から、「border="0"」「border属性なし」「border="5"」<br>
<a href="anywhere.html">
<img src="image.jpg" width="140" height="100" alt="" border="0"></a>
<a href="anywhere.html">
<img src="image.jpg" width="140" height="100" alt=""></a>
<a href="anywhere.html">
<img src="image.jpg" width="140" height="100" alt="" border="5"></a>
</p>
```

```
<h1>リンクしていない場合</h1>
<p>
※左から、「border属性なし」「border="2"」「border="5"」<br>
<img src="image.jpg" width="140" height="100" alt="">
<img src="image.jpg" width="140" height="100" alt="" border="2">
<img src="image.jpg" width="140" height="100" alt="" border="5">
</p>
```

 スタイルシート：「ボックス」の「枠線の太さを指定する」（P.241）



TIPS

### CSSで画像の枠を設定する場合の問題

Netscape Navigator 4.Xでは、画像の周りに表示させる枠線を、CSSで正しく設定することができません（枠線がおかしな部分に表示されます）。したがって、画像をリンクさせた時に表示される枠線を消したいような場合には、非推奨のborder属性を使うことも仕方がないかもしれません。

しかし、どうしても非推奨属性を使いたくない場合には、CSSで部分的にa要素の色を背景色と同じに設定して枠線を見えないようにするという裏ワザ的な対処法もあります。

# 画像とテキストの縦の位置関係を設定する

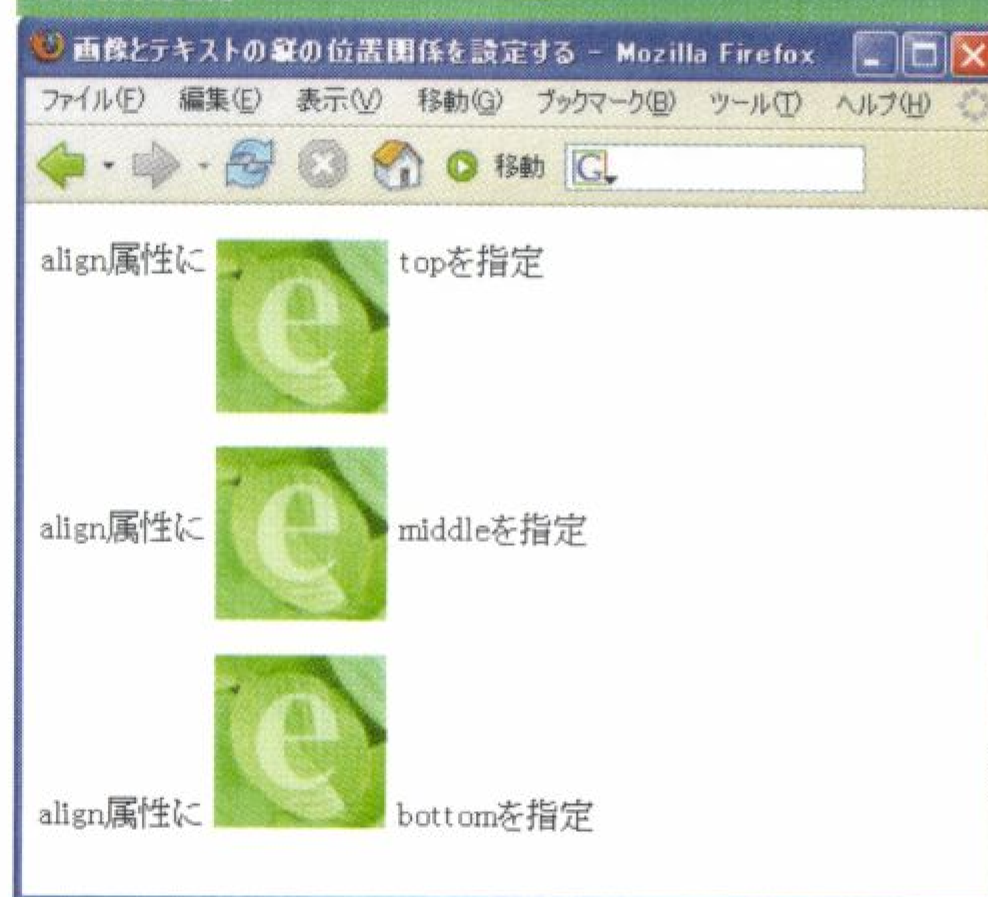
<img src="*URL*" alt="*代替テキスト*" align="*位置*">

| 【位置】 | |
|---|---|
| top | 画像の上とテキストの上を揃える |
| middle | 画像の中心とテキストのベースラインを揃える |
| bottom | 画像の下とテキストのベースラインを揃える(デフォルト) |

Internet Explorer 6.0




Firefox 1.0




align属性を使用すると、同じ行の中に画像とテキストが表示される場合の、画像とテキストの縦の位置関係を設定できます。
ただし、align属性を使うことは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。位置関係を設定する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<p>align属性に
<img src="leaf.gif" width="80" height="80" alt="" align="top">
topを指定
</p>
<p>align属性に
<img src="leaf.gif" width="80" height="80" alt="" align="middle">
middleを指定
</p>
<p>align属性に
<img src="leaf.gif" width="80" height="80" alt="" align="bottom">
bottomを指定
</p>
```


スタイルシート:「テキスト」の「縦方向の位置を指定する」(P.215)

# イメージマップを作成する

**<img src="*URL*" alt="*代替テキスト*" usemap="*#マップ名*">**
**<map name="*マップ名*">～</map>**
**<area shape="*形状*" coords="*座標*" href="*URL*" alt="*代替テキスト*">**

| | |
|---|---|
| マップ名 | イメージマップの名前 |
| 形状 | rect(四角形)・circle(円)・poly(多角形)・default(全体) |
| 座標 | 形状別の座標値 |

Internet Explorer 6.0



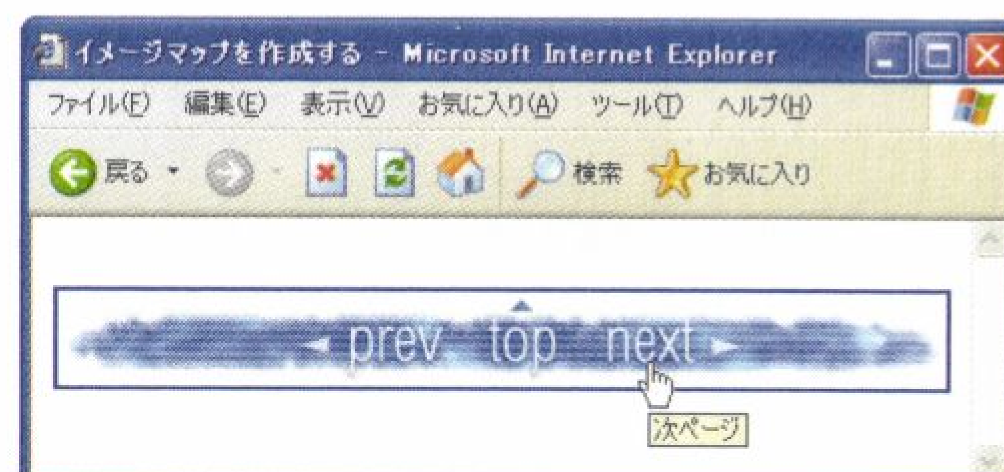

Firefox 1.0










イメージマップとは、画像の特定の領域がリンクになっているものです。map要素は、その内容がイメージマップの定義であることを示し、その定義部分に名前を付けます。map要素内で、実際にクリックに反応する領域とそのリンク先を設定するのがarea要

素です。area 要素には、リンク先を示す代替テキストを必ず指定しておいてください。後は、img 要素の usemap 属性で、定義したイメージマップの名前を指定すれば(名前の前には「#」を付けます)、画像がイメージマップとして機能するようになります。
area 要素の coords 属性で指定する座標の指定方法は、shape 属性で指定する領域の形状によって異なります。指定方法は次の通りで、各座標値はピクセル単位で「,」で区切って指定します。

**Sample**

```
<div>
<img src="nav.gif" usemap="#navbar" alt="メニュー" ↵
 width="422" height="44">
<map name="navbar">
<area href="prev.html" shape="rect" alt="前ページ " ↵
 coords="113,12,183,36">
<area href="top.html"  shape="rect" alt="トップページ " ↵
 coords="199,2,241,41">
<area href="next.html" shape="rect" alt="次ページ" ↵
 coords="258,12,327,36">
</map>
</div>
```



**コラム**

## coords 属性での座標の指定方法

| | |
|---|---|
| 四角形(rect)の場合 | 「左上のＸ座標」「左上のＹ座標」「右下のＸ座標」「右下のＹ座標」 |
| 円(circle)の場合 | 「中心のＸ座標」「中心のＹ座標」「半径」 |
| 多角形(poly)の場合 | すべての角の座標を「Ｘ座標」「Ｙ座標」の順で指定<br>(ただし、最初と最後は同じ座標を指定する必要があります) |

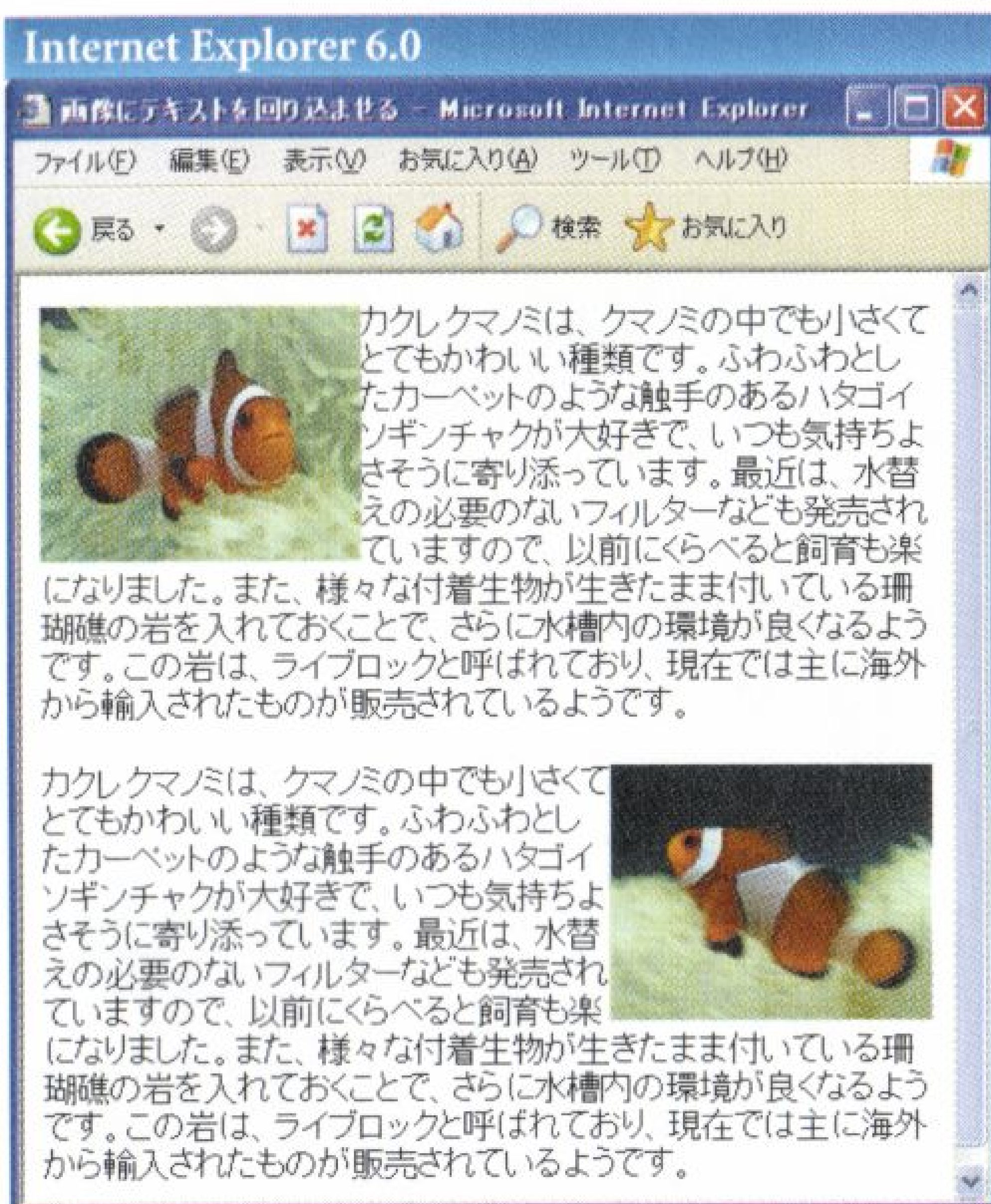
X=0,Y=0
X=113,Y=12
X=199,Y=2
X=258,Y=12
prev
top
next
X=183,Y=36
X=241,Y=41
X=327,Y=36
X=422,Y=44

# 画像にテキストを回り込ませる

## <img src="*URL*" alt="*代替テキスト*" align="*位置*">

| 位置 | left・right |
|---|---|

Internet Explorer 6.0




Firefox 1.0




img要素のalign属性で、画像を左または右に配置すると、その横にテキストを回り込ませます。
回り込みを解除したい場合には、br要素のclear属性を使用してください。
ただし、align属性を使うことは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。テキストを回り込ませる場合には、できるだけスタイルシートを利用するようにしてください。

### Sample

```
<p>
<img src="fish1.jpg" width="155" height="120" alt="" align="left">
カクレクマノミは、クマノミの中でも小さくてとてもかわいい種類です。（中略）この岩は、ライブ
ロックと呼ばれており、現在では主に海外から輸入されたものが販売されているようです。
<br clear="left">
</p>
<p>
<img src="fish2.jpg" width="155" height="120" alt="" align="right">
カクレクマノミは、クマノミの中でも小さくてとてもかわいい種類です。（中略）この岩は、ライブ
ロックと呼ばれており、現在では主に海外から輸入されたものが販売されているようです。
<br clear="right">
</p>
```

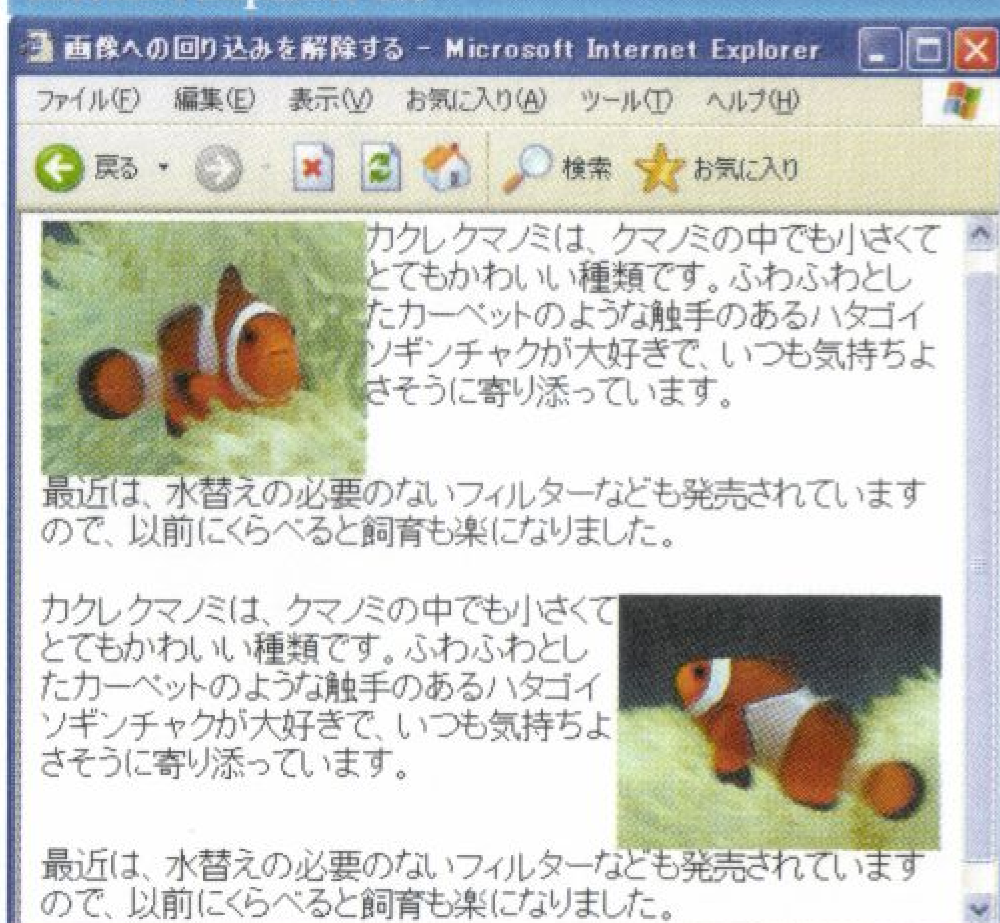
スタイルシート：「表示と配置」の「左右への配置と回り込みを指定する」（P.252）

# 画像への回り込みを解除する

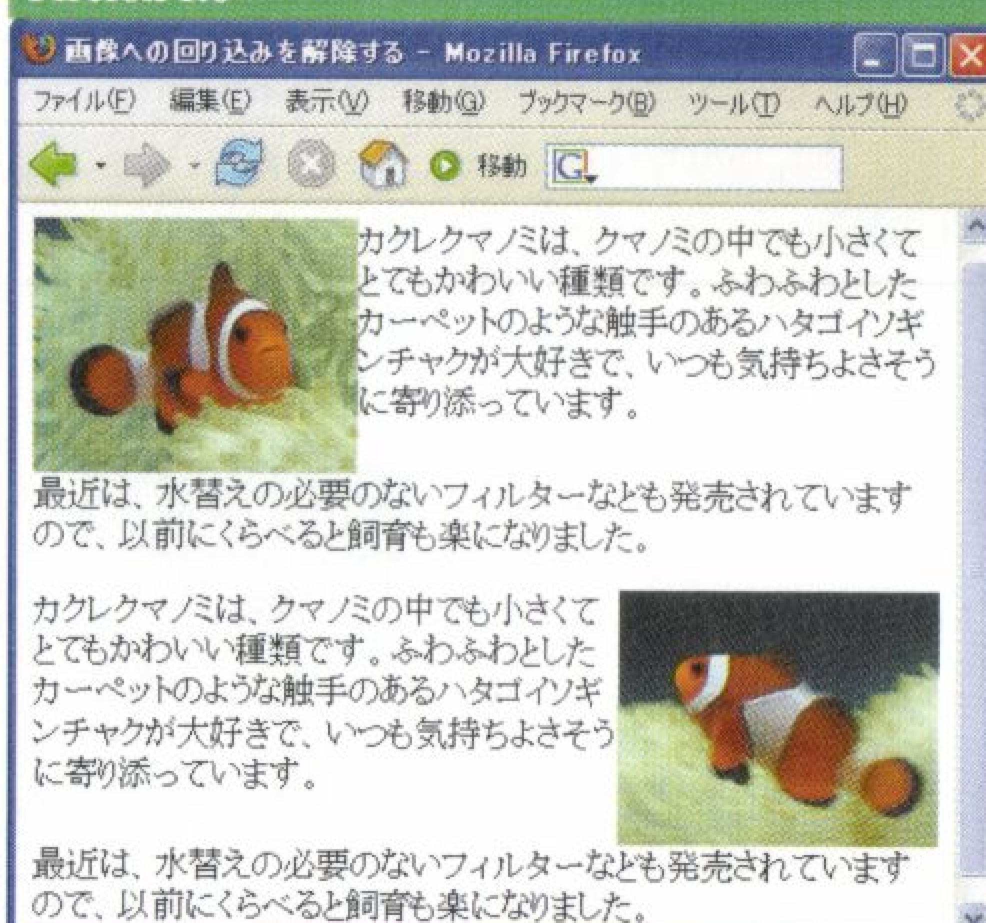
**<br clear="どちら側の画像の回り込みを解除するか">**

| 【どちら側の画像の回り込みを解除するか】 | |
|---|---|
| left | 左側の画像に対する回り込みを解除 |
| right | 右側の画像に対する回り込みを解除 |
| all | 両側の画像に対する回り込みを解除 |

Internet Explorer 6.0




Firefox 1.0




br要素のclear属性は、画像の横にテキストを回り込ませている状態を解除します。この指定以降のテキストは、画像の下の行から表示されるようになります。
ただし、clear属性を使うことは非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。テキストの回り込みを解除する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<p>
<img src="fish1.jpg" width="155" height="120" alt="" align="left">
カクレクマノミは、(中略)いつも気持ちよさそうに寄り添っています。
<br clear="left">
最近は、(中略)前にくらべると飼育も楽になりました。
</p>
<p>
<img src="fish2.jpg" width="155" height="120" alt="" align="right">
カクレクマノミは、(中略)いつも気持ちよさそうに寄り添っています。
<br clear="right">
最近は、(中略)以前にくらべると飼育も楽になりました。
</p>
```

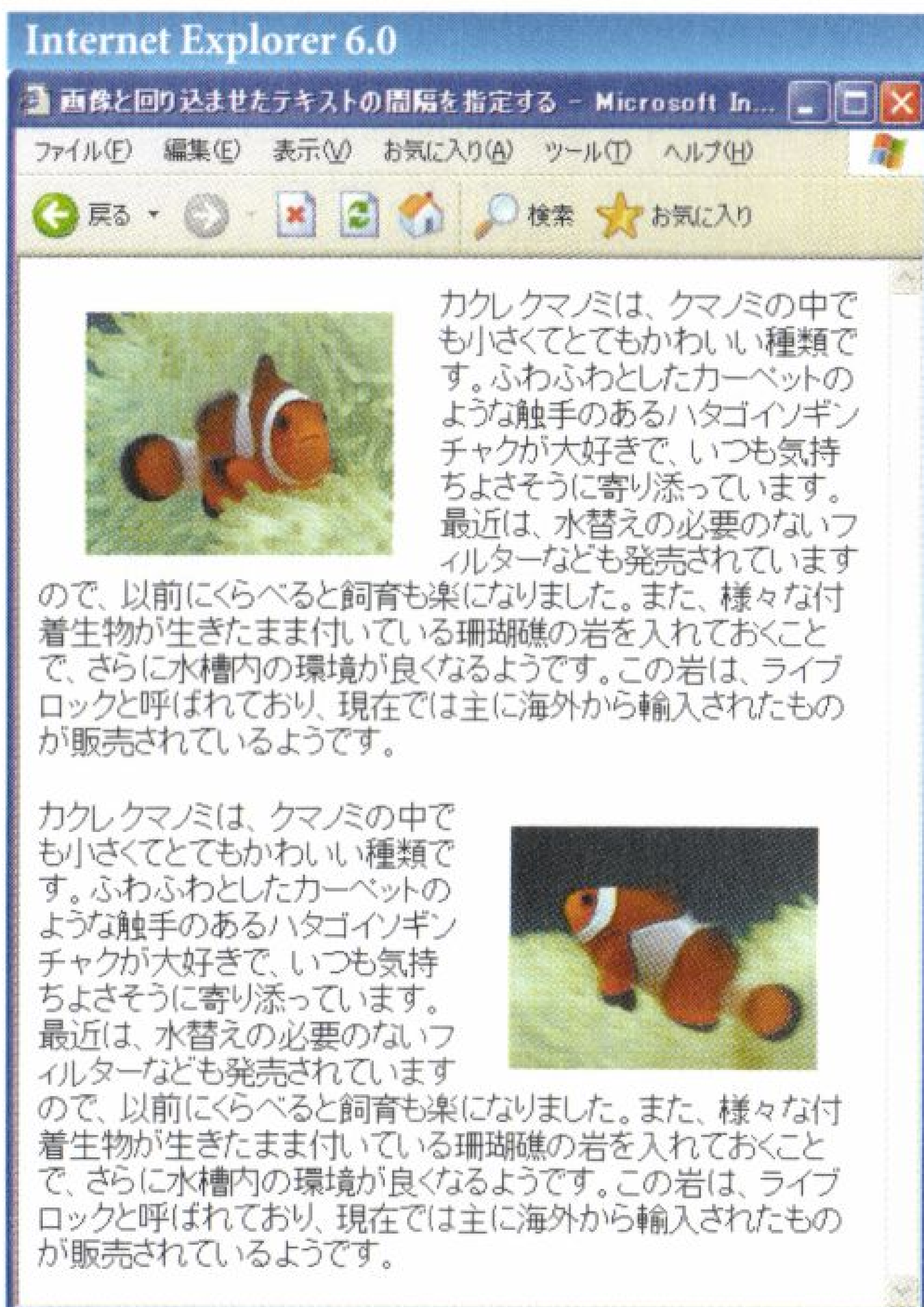

スタイルシート:「表示と配置」の「回り込みを解除する」(P.254)

# 画像と回り込ませたテキストの間隔を指定する

**<img src="*URL*" alt="*代替テキスト*" vspace="*縦間隔*" hspace="*横間隔*">**

| | |
|---|---|
| 縦間隔 | 画像の上下の間隔（ピクセル） |
| 横間隔 | 画像の左右の間隔（ピクセル） |

Internet Explorer 6.0




Firefox 1.0




画像の横にテキストを回り込ませた場合の、画像とテキストとの間隔を設定します。vspace属性を指定すると画像の上下に、hspace属性を指定すると画像の左右に指定したピクセル数の空間ができます。
ただし、vspace属性とhspace属性はいずれも非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。間隔を空けたい場合には、できるだけスタイルシートのマージンなどで調節するようにしてください。

## Sample

```
<p>
<img src="fish1.jpg" width="155" height="120" alt="" align="left"
 vspace="12" hspace="24">
```

```
カクレクマノミは、クマノミの中でも小さくてとてもかわいい種類です。ふわふわとしたカーペットのような触手のあるハタゴイソギンチャクが大好きで、いつも気持ちよさそうに寄り添っています。最近は、水替えの必要のないフィルターなども発売されていますので、以前にくらべると飼育も楽になりました。また、さまざまな付着生物が生きたまま付いている珊瑚礁の岩を入れておくことで、さらに水槽内の環境が良くなるようです。この岩は、ライブロックと呼ばれており、現在では主に海外から輸入されたものが販売されているようです。
<br clear="left">
</p>
<p>
<img src="fish2.jpg" width="155" height="120" alt="" align="right" vspace="12" hspace="24">
カクレクマノミは、クマノミの中でも小さくてとてもかわいい種類です。ふわふわとしたカーペットのような触手のあるハタゴイソギンチャクが大好きで、いつも気持ちよさそうに寄り添っています。最近は、水替えの必要のないフィルターなども発売されていますので、以前にくらべると飼育も楽になりました。また、さまざまな付着生物が生きたまま付いている珊瑚礁の岩を入れておくことで、さらに水槽内の環境が良くなるようです。この岩は、ライブロックと呼ばれており、現在では主に海外から輸入されたものが販売されているようです。
<br clear="right">
</p>
```


スタイルシート：「ボックス」の「マージンを設定する」(P.237)



**コラム**

### alt属性には適切な代替テキストを！

テキストブラウザや音声ブラウザなどでさまざまなホームページを見て（読み上げて）いると、時々意味がわからなくなることがあります。そして、その原因のうち、もっとも多いと思われるのが、画像の代替テキストの不備です。

代替テキストがなければ当然その部分の情報は欠落しますし、特に画像がリンクとして使用されている場合には、リンク自体が利用できなくなったり、何のリンクかがわからない状態となります。また、代替テキストが付けられていても、その内容が不適切であれば、かえって意味がわからなくなることもあります。

画像に代替テキストを付けるのはもちろんですが、その内容を考える場合には「もし、画像を配置することをやめて、その部分をテキストで表現するとしたらどう書くか」を想像してみると、適切なものが思い浮かぶかもしれません。

# さまざまな形式のデータを配置する

**<object data="*URL*" type="*MIMEタイプ*" width="*幅*" height="*高さ*">～</object>**

| | |
|---|---|
| URL | 配置するデータのURL |
| MIMEタイプ | 配置するデータのMIMEタイプ |
| 幅 | 配置するデータの幅(ピクセル·%) |
| 高さ | 配置するデータの高さ(ピクセル·%) |

object要素は、さまざまな形式のデータを配置することのできる汎用的な要素です。
具体的には、画像·動画・JAVAアプレット・プラグインデータ・他のHTML文書などを配置することができます。この要素の特徴は、指定した形式のデータをブラウザが取り扱うことができる場合には、<object>～</object>の間の内容を無視することです(param要素とmap要素は除く)。したがって、配置させたいデータ形式順にobject要素を入れ子にしておくと、ブラウザが利用可能なデータ形式があった時点でそれを配置して、さらに深い入れ子となっている別の形式を無視することになります。
ただし、現在のところ、この要素に正しく対応しているブラウザは多くはないようです。

※この要素は多くの種類のデータ形式をサポートしているため、実際にはさらに多くの属性がありますが、ここでは省略しています。

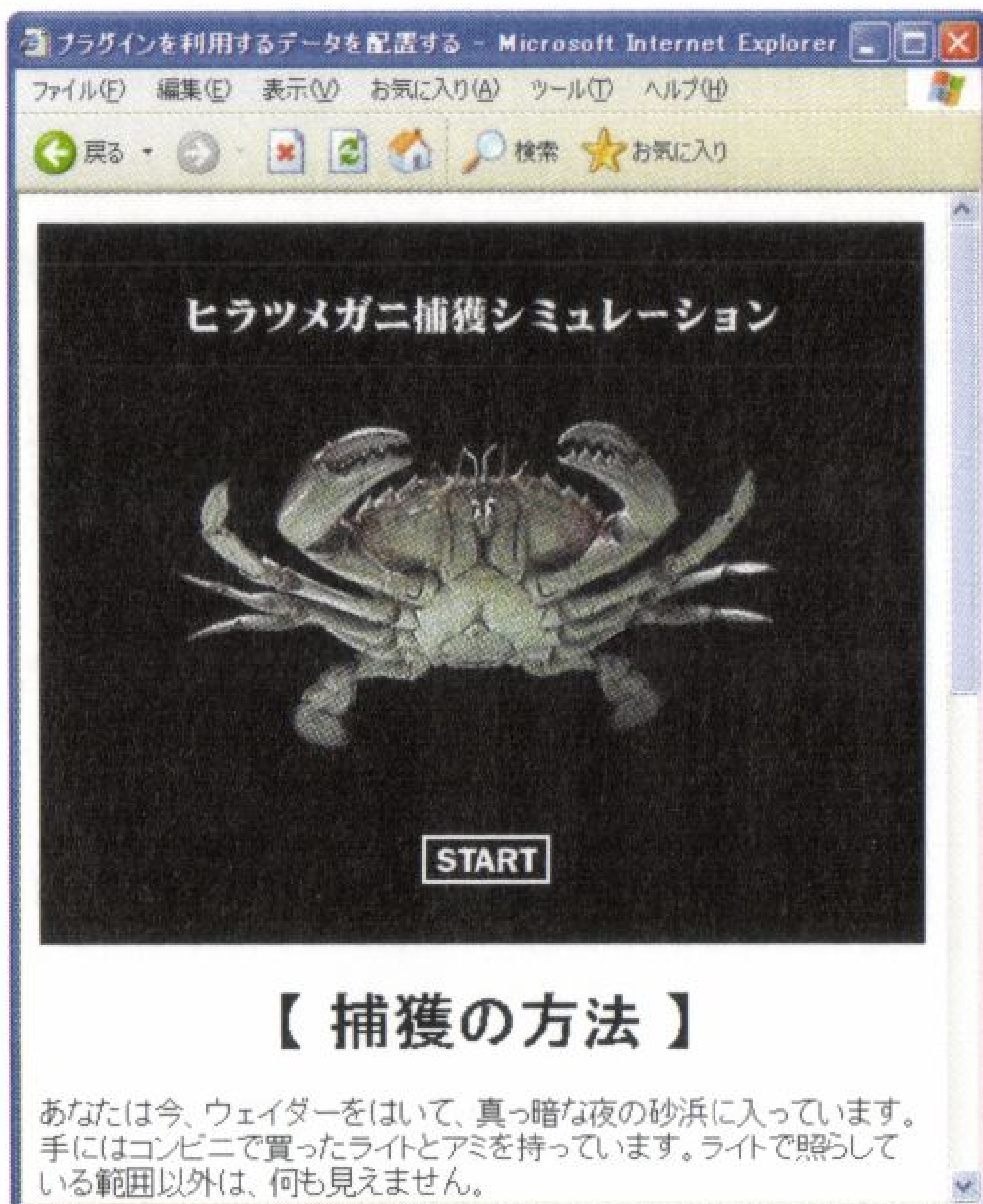
## Sample

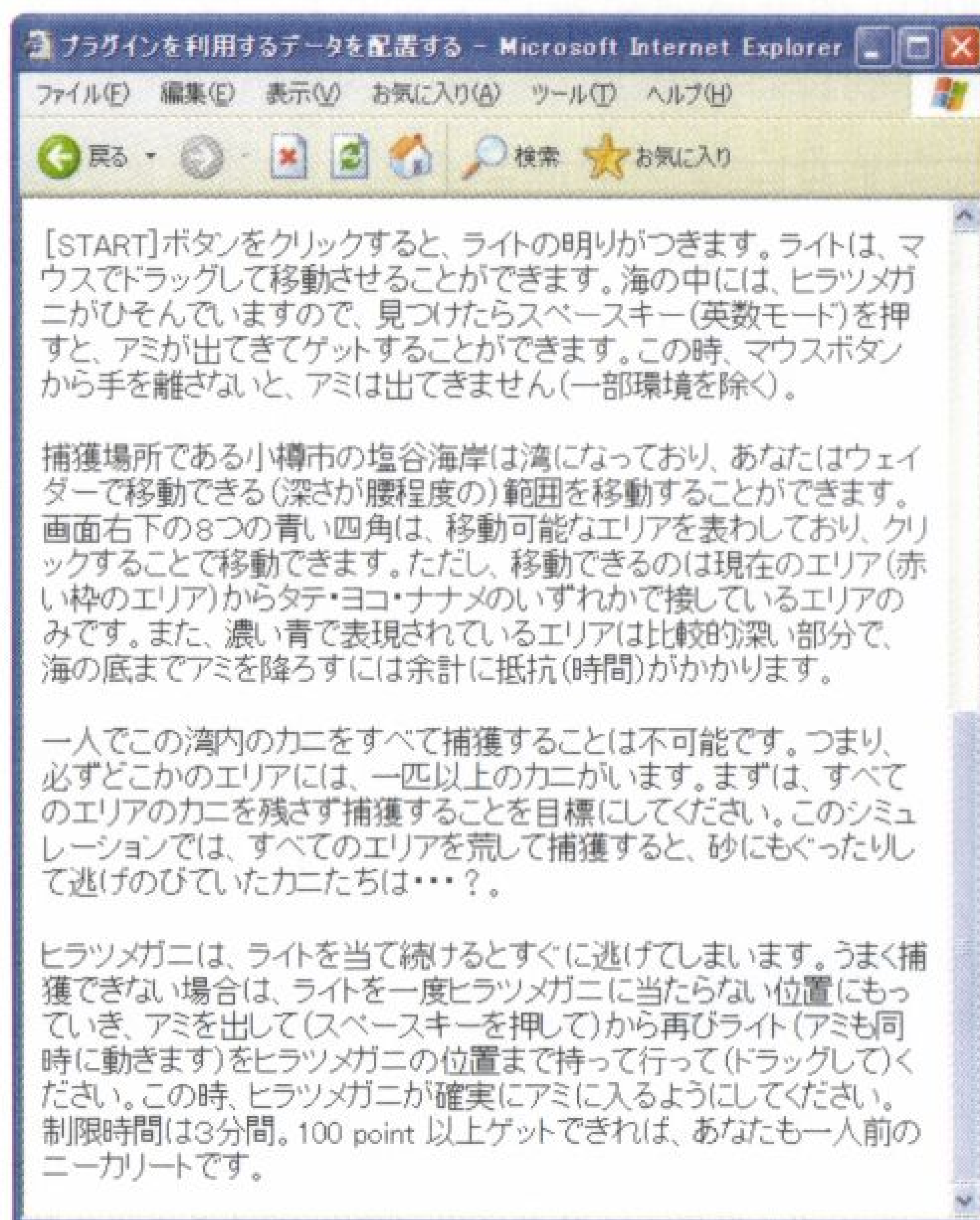
```
<p>
<object data="fireworks.mpeg" type="application/mpeg">
<object data="fireworks.gif" type="image/gif">
色鮮やかな花火が印象的だった。
</object>
</object>
</p>
```

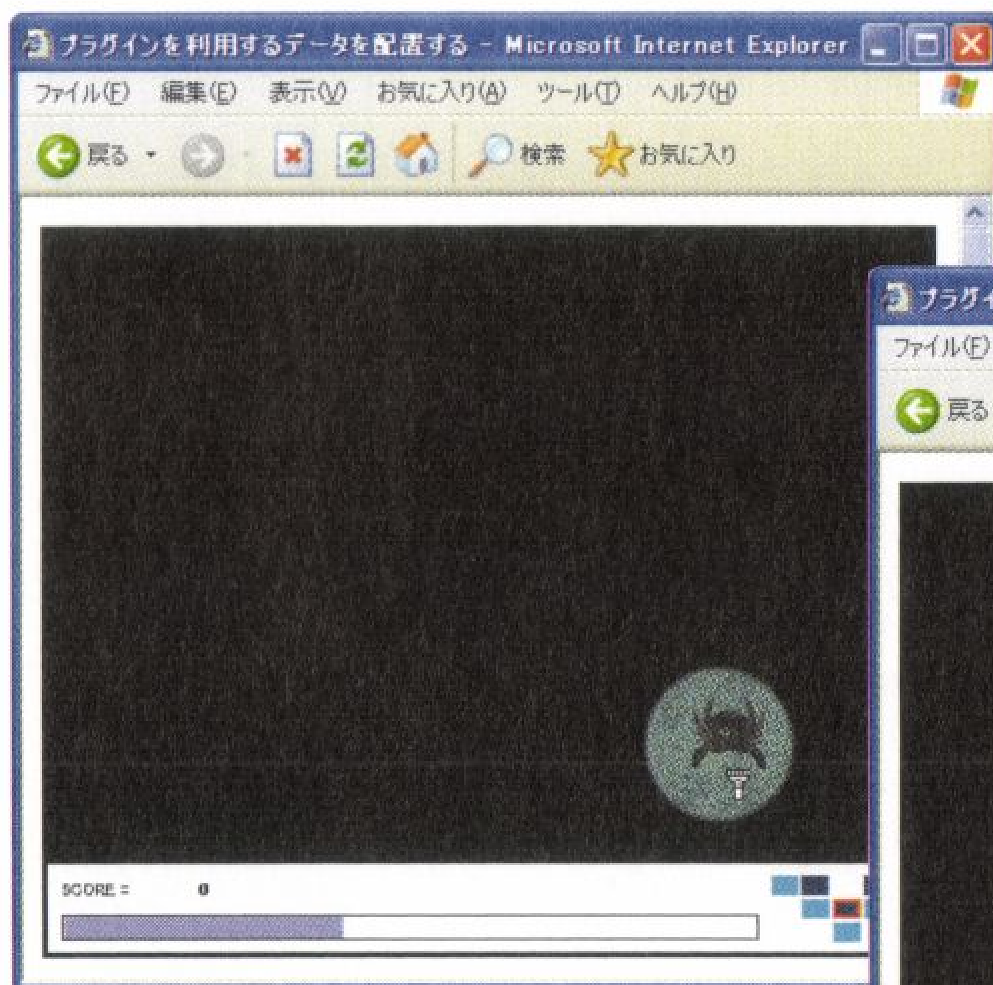
# プラグインを利用するデータを配置する

**<embed src="*URL*" width="*幅*" height="*高さ*">～</embed>**
**<noembed>～</noembed>**

| | |
|---|---|
| URL | プラグインが使用するデータのURL |
| 幅 | ピクセル |
| 高さ | ピクセル |






embed要素は、プラグイン機能を利用する場合に使用する要素です。
この要素は、object要素が定義される以前に一部のブラウザが独自に採用したもので、標準的な仕様では定義されていないものです。
基本的には、上の書式で紹介した属性が利用できますが、使用するプラグインによって他のさまざまな属性が利用可能です。詳しくは、各プラグインのマニュアルなどを参照してください。
noembed要素には、プラグインが利用できない場合に表示させたい内容を入れておきます。ここで指定した内容は、プラグインが利用可能な場合には表示されません。

## Sample

```
<div>
<embed src="crab98.dcr" width="464" height="372">
<noembed>
このシミュレーションを実行するためには、<a href="http://www.macromedia.com/jp/downloads/">Macromedia Shockwave Player</a>が必要です。<a href="aboutsim.html">シミュレーションの内容を紹介</a>しているページもあります。
</noembed>
</div>
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac



# JAVAアプレットを配置する

**<applet code="クラスファイル名" width="幅" height="高さ">～</applet>**
**<param name="パラメータ名" value="パラメータの値">**

| | |
|---|---|
| 幅 | ピクセル･% |
| 高さ | ピクセル･% |

applet要素は、JAVAアプレットを利用する場合に使用する要素です。
実際には、上の書式で紹介した属性以外にも利用可能な属性がありますが、ここでは省略します。<applet>～</applet>の範囲には、JAVAアプレットが利用できない場合に表示させたい内容を入れておきます。ここで指定した内容は、JAVAが動作可能な場合には表示されません。
param要素を利用すると、JAVAアプレットを実行する時に必要な値を指定しておくことができます。この場合、param要素は<applet>～</applet>の範囲の最初に記述してください。
なお、applet要素を使うことは非推奨とされており、代わりにobject要素を使うことが推奨されています。JAVAについての詳細は、専門書を参照してください。

## Sample

```
<p>
<applet code="colorchart.class" width="475" height="400">
JAVAが利用できない環境の方は、<a href="colorchart2.html">
カラーチャート2</a>をご利用ください。
</applet>
</p>
```

# フォームを作る

**<form action="*URL*" method="*送信形式*" enctype="*MIMEタイプ*" target="*ウィンドウ名*">～</form>**

| | |
|---|---|
| URL | 送信されたフォームを処理するプログラムのURL |
| 送信形式 | get(デフォルト)・post |
| MIMEタイプ | post形式で内容を送信する際のMIMEタイプ |
| ウィンドウ名 | 送信した結果を表示するウィンドウまたはフレーム名 |




form要素は、その内容が送信可能な入力フォームであることを示します。
入力された内容の送り先や送信方法についても、この要素の属性で指定します。<form>～</form>の範囲には、まずp要素などのブロックレベル要素を配置して、その中に入力や選択に必要となる要素(input要素・button要素・select要素・textarea要素など)を入れるようにしてください。内容として用意した送信ボタン(input要素かbutton要素で「type="submit"」が指定されているもの)が押されると、フォームの内容が送信されます。
送信形式には、「get」と「post」の2種類があります。「get」を指定すると、入力されたデータは、action属性で指定されているURLの後に「?」記号で連結された状態で送信されます。この形式は、サーチエンジンなどでよく利用されています。「post」は、送信する内容が日本語でデータ量が多い場合などに使用される形式で、入力された内容をURLに連結することなく、データとして送信します。
enctype属性で指定するMIMEタイプは、デフォルトでは「application/x-www-form-urlencoded」になっています。通常はこれを利用してCGIで処理を行いますが、input要素の「type="file"」でファイルを送信するような場合には、MIMEタイプに「multipart/form-data」を指定してください。

form要素は、その内容を送信することを前提としているため、送信先を示すaction属性は必ず指定することになっています。内容を送信する必要がない場合には、form要素を使用せずに入力フィールドやボタン、メニューなどを直接配置してください(ただし、Netscape Navigator 4.Xの場合は、form要素の中に配置しなければ表示されなくなります)。

## Sample

```
<form action="/cgi-bin/formmail.cgi" method="post">
<p>
<label for="nm">お名前：</label>
<input type="text" name="namae" id="nm"><br>
<label for="ma">メール：</label>
<input type="text" name="email" id="ma">
</p>
<p>
<input type="radio" name="sex" value="male" id="sm">
<label for="sm">男性</label>
<input type="radio" name="sex" value="female" id="sf">
<label for="sf">女性</label>
</p>
<p>
<input type="submit" value="送信">
<input type="reset" value="リセット">
</p>
</form>
```


<label for="～">：「フォーム」の「ラベルテキストと項目を一体化させる」(P.132)
<form action="mailto:～">：「フォーム」の「フォームの内容がメールで届くようにする」(P.133)

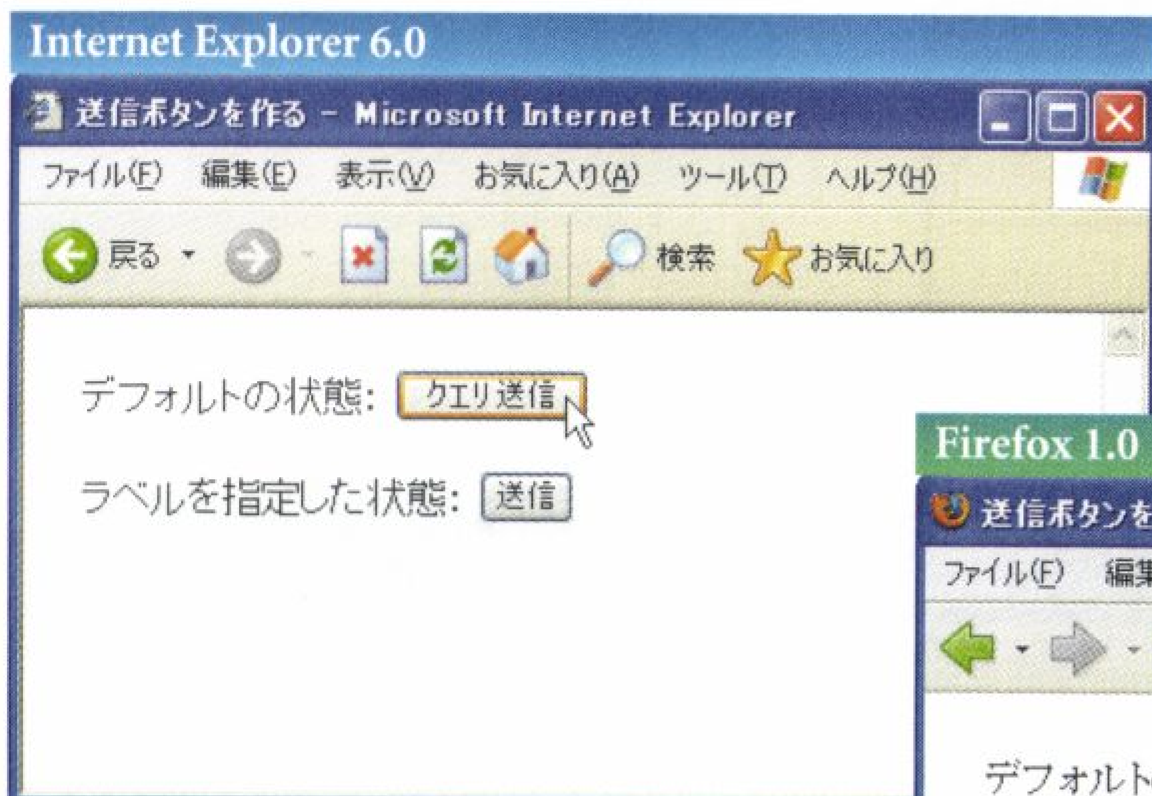
# 送信ボタンを作る

## <input type="submit" value="ラベル" name="名前">

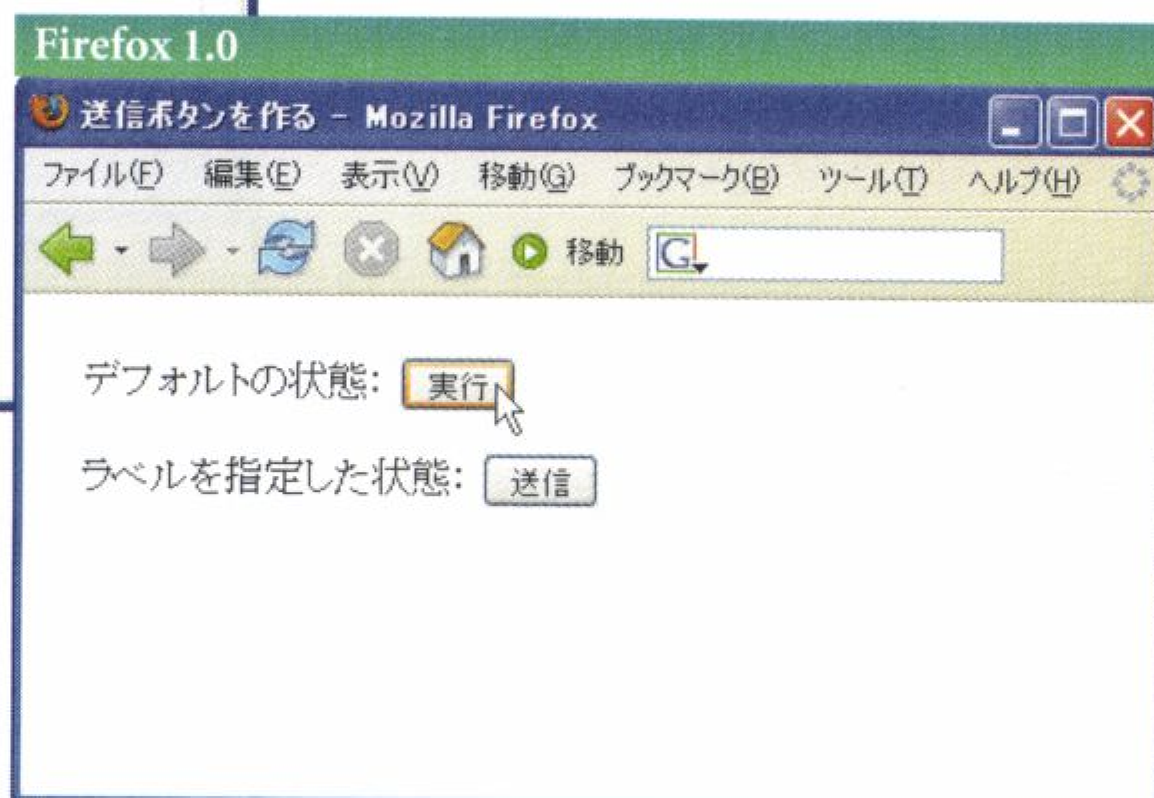
| | |
|---|---|
| ラベル | ボタン上に表示される文字 |
| 名前 | ボタンの名前 |



Internet Explorer 6.0




Firefox 1.0




input要素の「type="submit"」で、form要素の設定にしたがって、フォーム内のデータを送信するためのボタンを作成します。
value属性を指定すると、ボタン上に表示される文字を指定することができます。特に指定しなかった場合には、ブラウザがデフォルトの文字を表示させます(デフォルトの文字はブラウザによって異なります)。別の処理をさせたい複数の送信ボタンを配置する場合には、name属性でそれぞれ別の名前を付けておくことで判別できるようになります。

## Sample

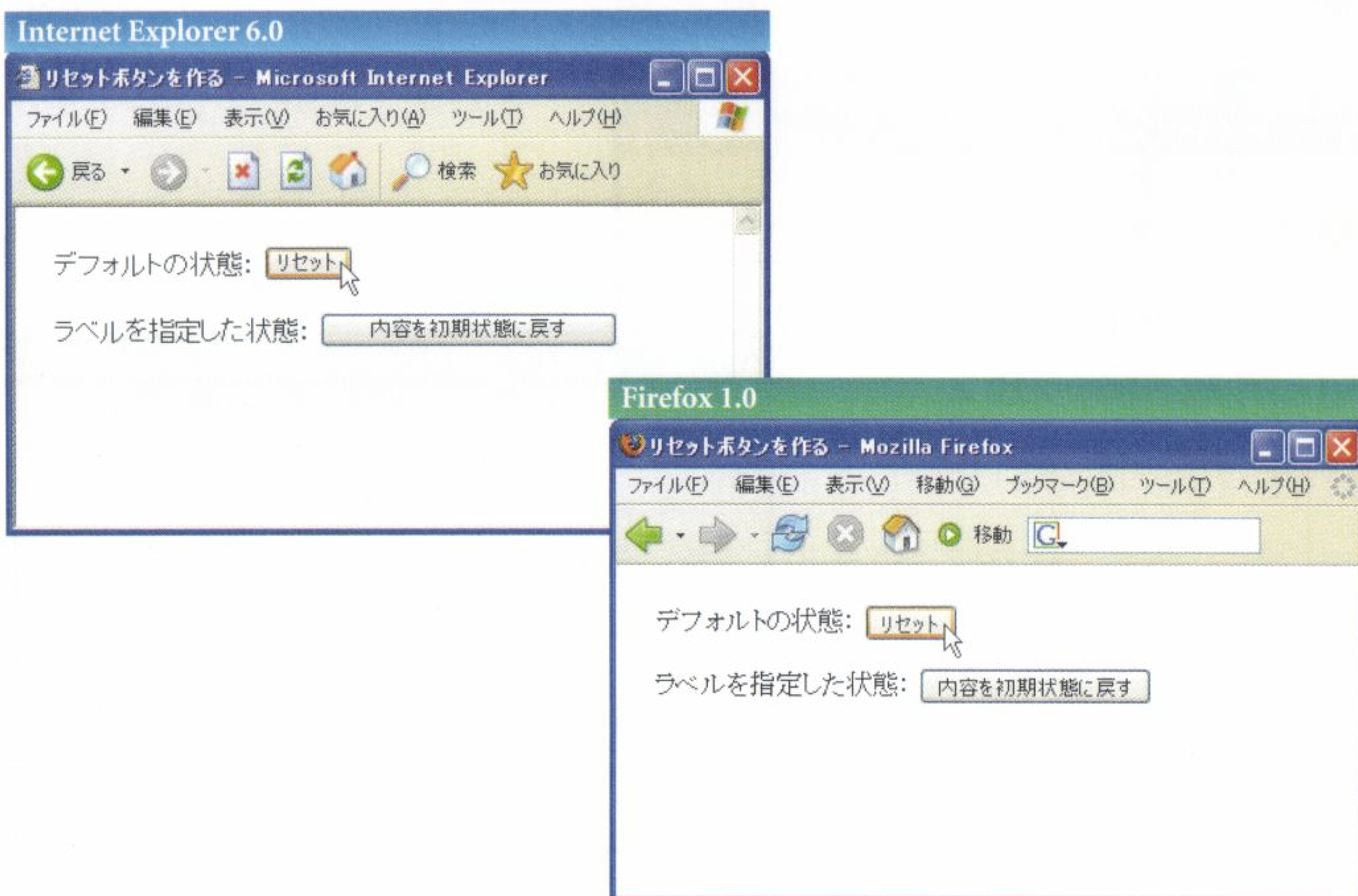
```
<p>
デフォルトの状態：
<input type="submit">
</p>
<p>
ラベルを指定した状態：
<input type="submit" value="送信">
</p>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac i-mode

# リセットボタンを作る

## <input type="reset" value="ラベル">

| | |
|---|---|
| ラベル | ボタン上に表示される文字 |




input要素の「type="reset"」は、フォームのすべての内容を初期値に戻すためのボタンを作成します。

value属性を指定すると、ボタン上に表示される文字を指定することができます。特に指定しなかった場合には、ブラウザがデフォルトの文字を表示させます(デフォルトの文字はブラウザによって異なります)。

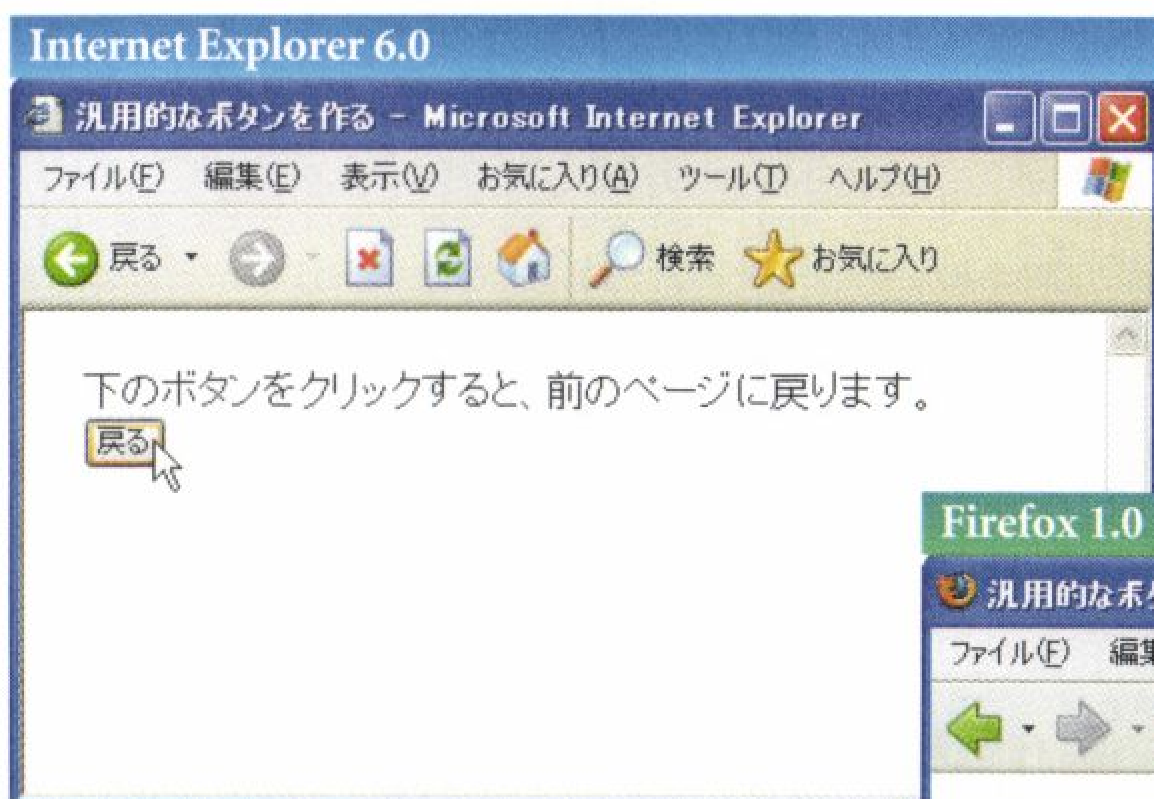
### Sample

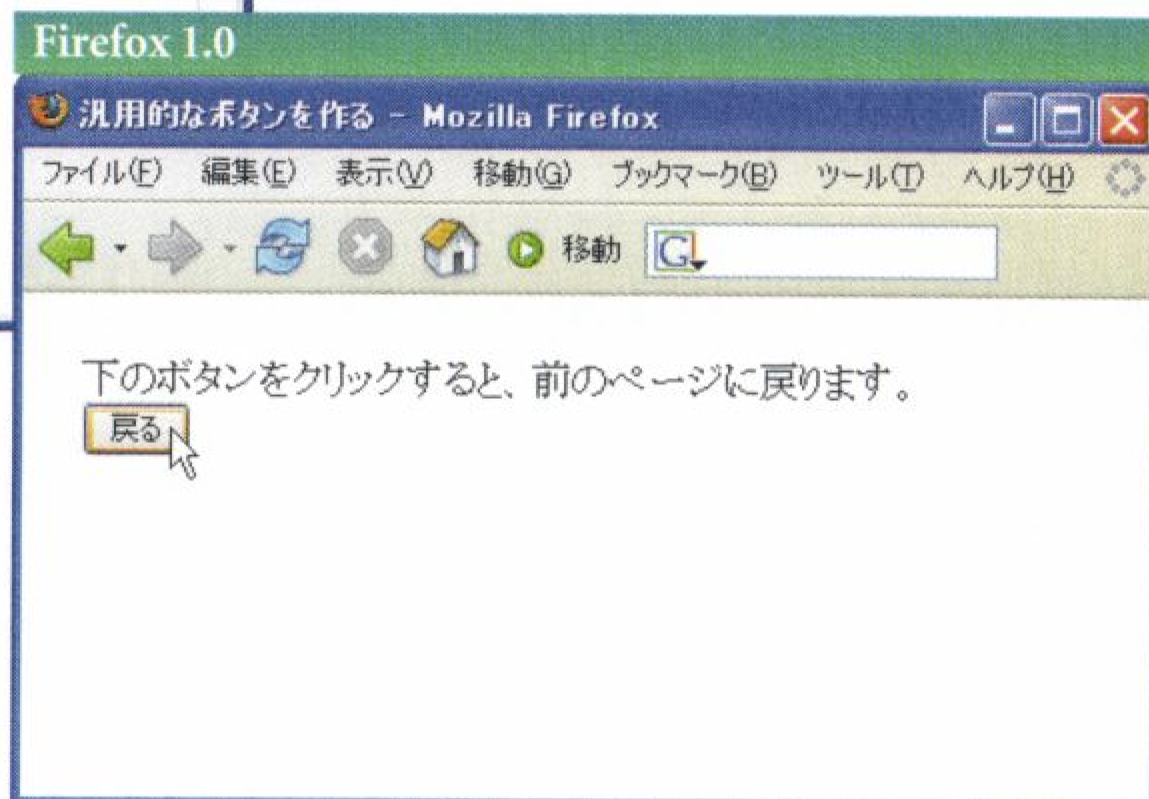
```
<p>
デフォルトの状態：
<input type="reset">
</p>
<p>
ラベルを指定した状態：
<input type="reset" value="内容を初期状態に戻す">
</p>
```

# 汎用的なボタンを作る

**<input type="button" name="名前" value="ラベル">**

| | |
|---|---|
| 名前 | ボタンの名前 |
| ラベル | ボタン上に表示される文字 |




input要素の「type="button"」で、送信もリセットもしない汎用のボタンを作成します。一般的には、onClickなどのイベント属性を利用して、JavaScriptなどのスクリプト言語と組み合わせて使用されます。

## Sample

```
<p>
下のボタンをクリックすると、前のページに戻ります。<br>
<input type="button" value="戻る" onClick="history.go(-1)">
</p>
```

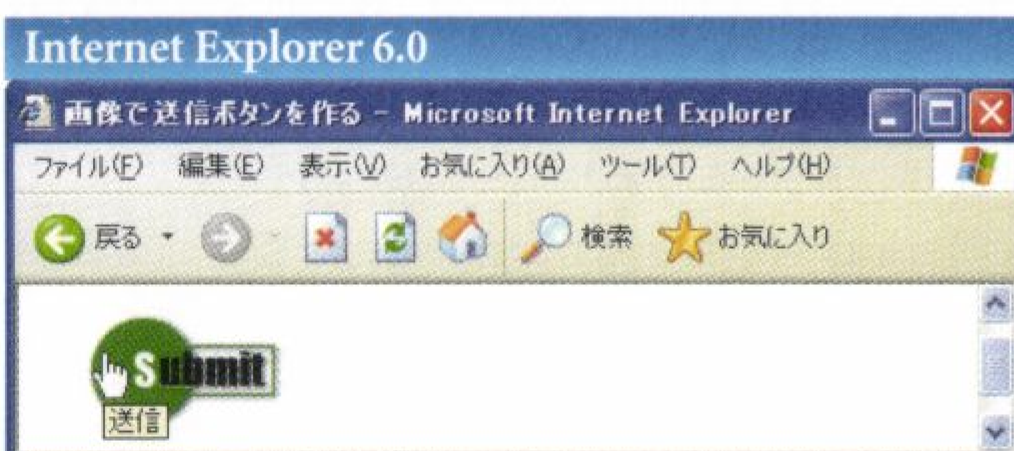
IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 画像で送信ボタンを作る

**<input type="image" src="*URL*" name="*名前*" alt="*代替テキスト*" align="*位置*">**

| | |
|---|---|
| URL | ボタンとして使用する画像ファイルのURL |
| 名前 | ボタンの名前 |
| 代替テキスト | 画像の代わりとなるテキスト |
| 【位置】 | |
| top | 画像の上とテキストの上を揃える |
| middle | 画像の中心とテキストのベースラインを揃える |
| bottom | 画像の下とテキストのベースラインを揃える（デフォルト） |
| left | 画像を左に配置し、右側に文字を回り込ませる |
| right | 画像を右に配置し、左側に文字を回り込ませる |

Internet Explorer 6.0




Firefox 1.0




通常、送信ボタンには<input type="submit">を使用しますが、画像を送信ボタンとして機能させることもできます。その場合、フォームの内容と共に画像のクリックされた位置（座標）も送信されます。
画像を送信ボタンとして使用する場合には、画像が表示されなくても送信ができるように、必ずalt属性で代替テキストを指定しておくようにしてください。ただし、古いブラウザの中には、name属性やvalue属性の値を代替テキストとして使用するものもあるようです。align属性は、通常の画像の場合と同様に表示位置を設定します。テキストが回り込むように設定した場合は、<br clear="*解除方向*">で解除することができます。
ただし、align属性やclear属性は非推奨とされており、新しいHTMLの標準仕様では使うことができなくなっています。テキストを回り込ませたり、それを解除する場合には、できるだけスタイルシートを利用するようにしてください。

## Sample

```
<p>
<input type="image" src="submit.gif" name="submit" alt="送信">
</p>
```

<br clear="～">：「画像とマルチメディア」の「画像への回り込みを解除する」(P.106)
スタイルシート：「表示と配置」の「左右への配置と回り込みを指定する」(P.252)
スタイルシート：「表示と配置」の「回り込みを解除する」(P.254)

フォームを作る

# 自由にデザインできるボタンを作る

**<button type="タイプ" name="名前" value="送信値">～</button>**

| | |
|---|---|
| 名前 | ボタンの名前 |
| 送信値 | ボタンの名前とともに送信される値 |
| 【タイプ】 | |
| submit | 送信ボタン(デフォルト) |
| reset | リセットボタン |
| button | 汎用ボタン |



Internet Explorer 6.0




Firefox 1.0




button要素は、ボタン作成専用の要素です。
type属性で指定する値によって、送信ボタン・リセットボタン・汎用ボタンのそれぞれの機能を果たすことができます。また、このボタンの場合は、<button>～</button>の範囲に配置された内容(テキスト関連の要素や画像など)をボタンのラベルとして表示させせることができます。name属性とvalue属性で示す値は、別の処理をさせたい複数の送信ボタンを配置する場合に、受信側でどの送信ボタンが押されたかを見分けるためなどに利用されます。

## Sample

```
<p>
<button type="submit" name="submit" value="new">
<img src="new.gif" width="65" height="50" alt=""><br>
<strong>新規</strong>
</button>
<button type="submit" name="submit" value="modify">
<img src="modify.gif" width="65" height="50" alt=""><br>
<strong>変更</strong>
</button>
</p>
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac
i-mode

# 1行の入力フィールドを作る

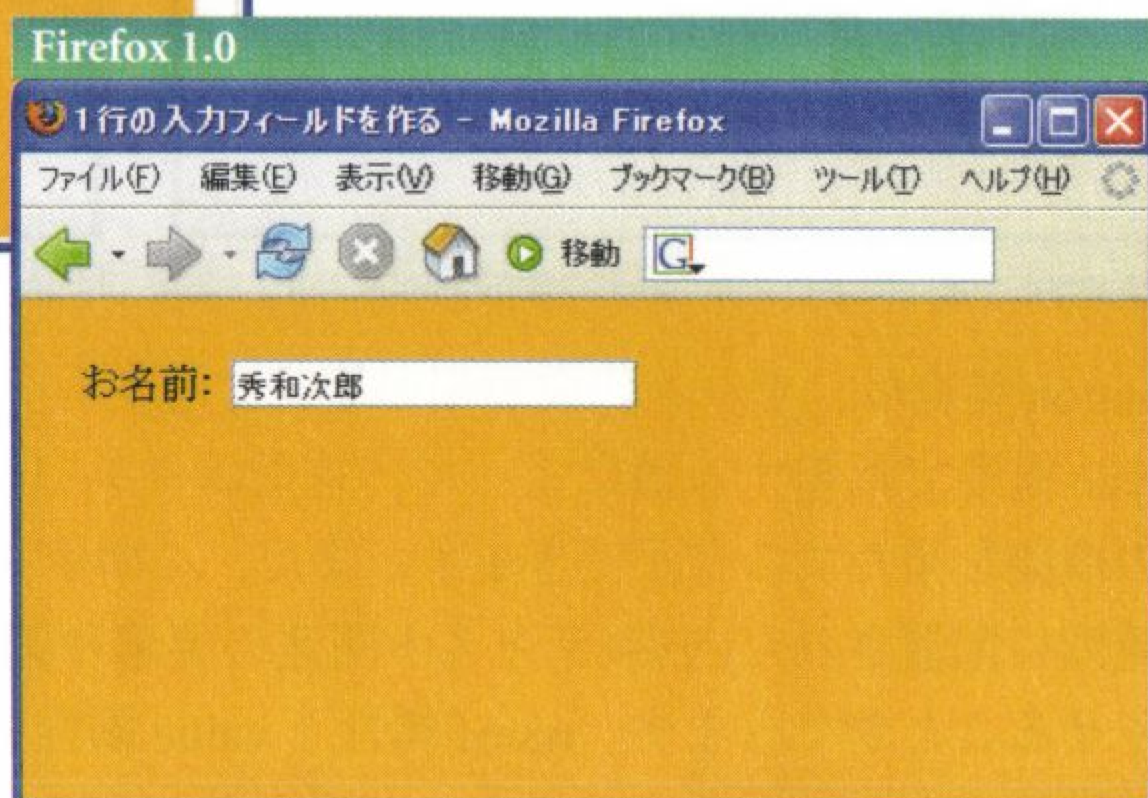
<input type="text" name="名前" value="デフォルト文字" size="幅" maxlength="最大文字数">

| | |
|---|---|
| 名前 | 入力フィールドの名前 |
| デフォルト文字 | あらかじめ入力されている文字 |
| 幅 | 入力フィールドの幅(文字数) |
| 最大文字数 | 入力可能な最大の文字数 |

Internet Explorer 6.0




Firefox 1.0




input要素の「type="text"」で、1行のテキスト入力フィールドを作成します。name属性で指定する名前は、フォームの内容を受信した時にデータを見分けるためなどに使用されます。

## Sample

```
<p>
<label for="nm">お名前：</label>
<input type="text" name="namae" size="30" id="nm">
</p>
```

<label for="〜">：「フォーム」の「ラベルテキストと項目を一体化させる」(P.132)

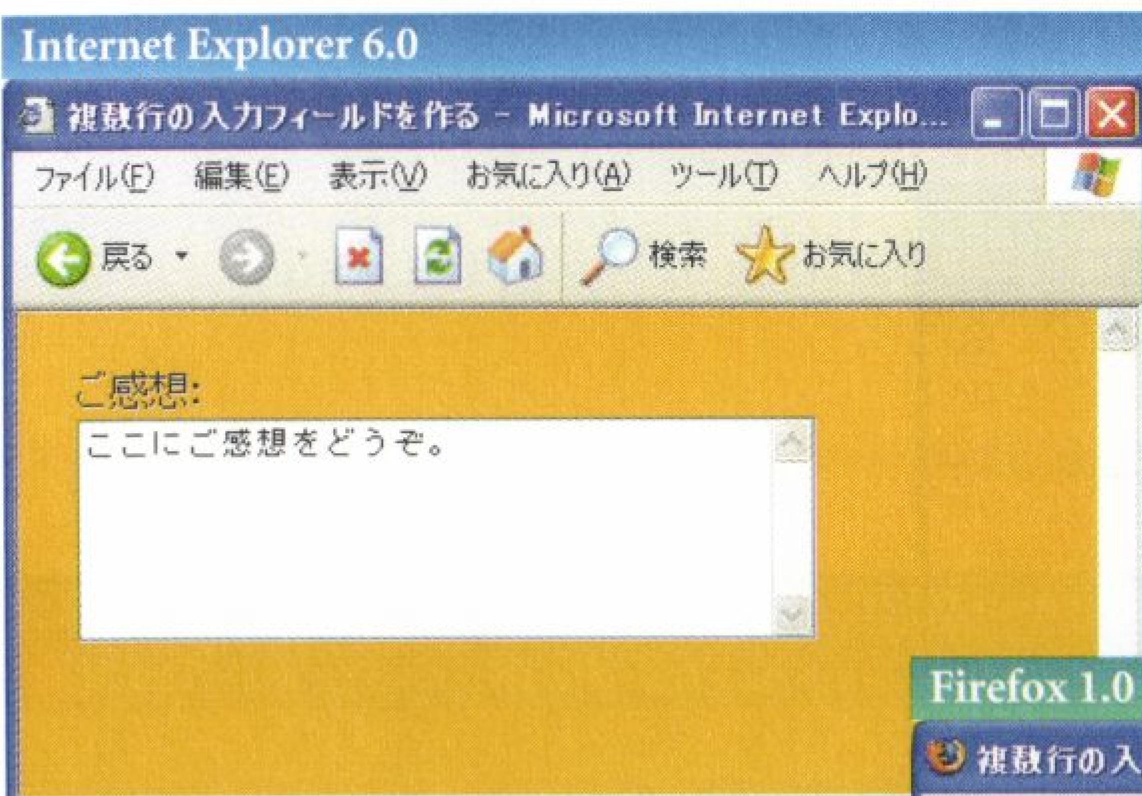
フォームを作る

# 複数行の入力フィールドを作る

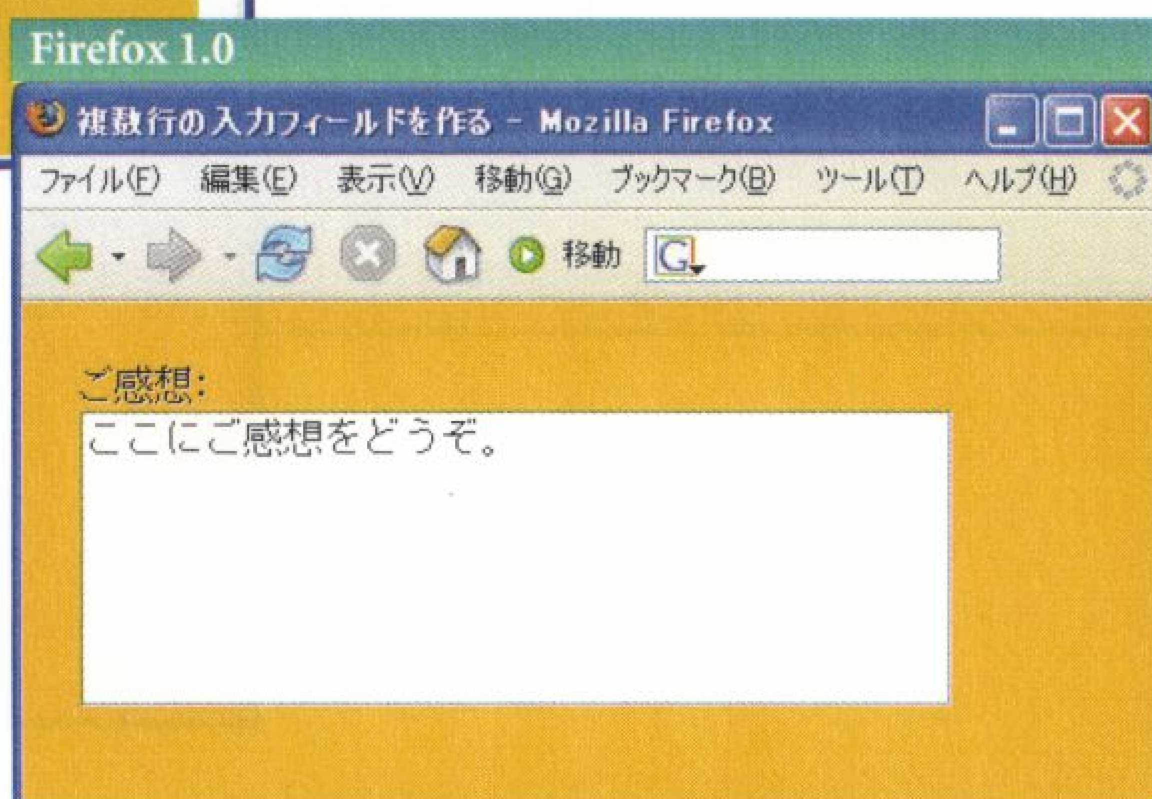
**<textarea name="名前" rows="行数" cols="幅">～</textarea>**

| | |
|---|---|
| 名前 | 入力フィールドの名前 |
| 行数 | 入力フィールドの行数 |
| 幅 | 入力フィールドの幅（文字数） |

Internet Explorer 6.0




Firefox 1.0




textarea要素は、複数行のテキスト入力フィールドを作成します。
<textarea>～</textarea>の範囲に記述した文字は、このフィールドにあらかじめ入力された状態で表示されます。name属性で指定する名前は、フォームの内容を受信した時にデータを見分けるためなどに使用されます。rows属性とcols属性は入力フィールドの大きさを決定するもので、必ず指定することになっています。

## Sample

```
<p>
ご感想：<br>
<textarea name="kansou" rows="6" cols="40">ここにご感想をどうぞ。
</textarea>
</p>
```

 <label for="～">：「フォーム」の「ラベルテキストと項目を一体化させる」（P.132）

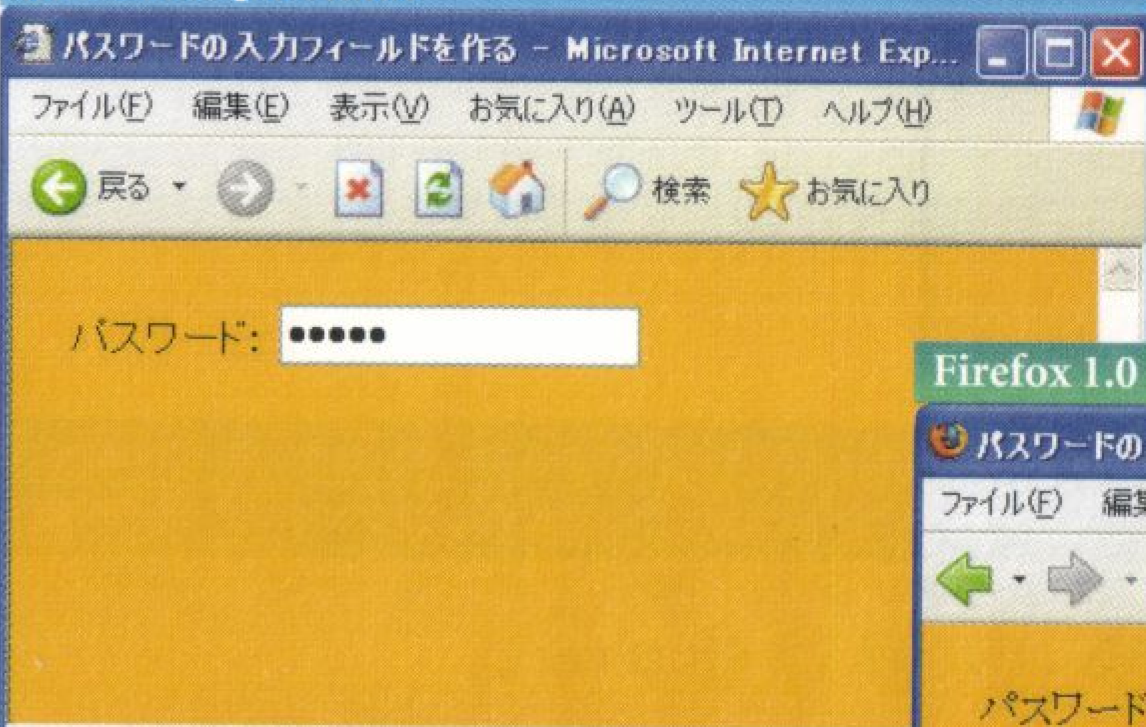
IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac i-mode

# パスワードの入力フィールドを作る

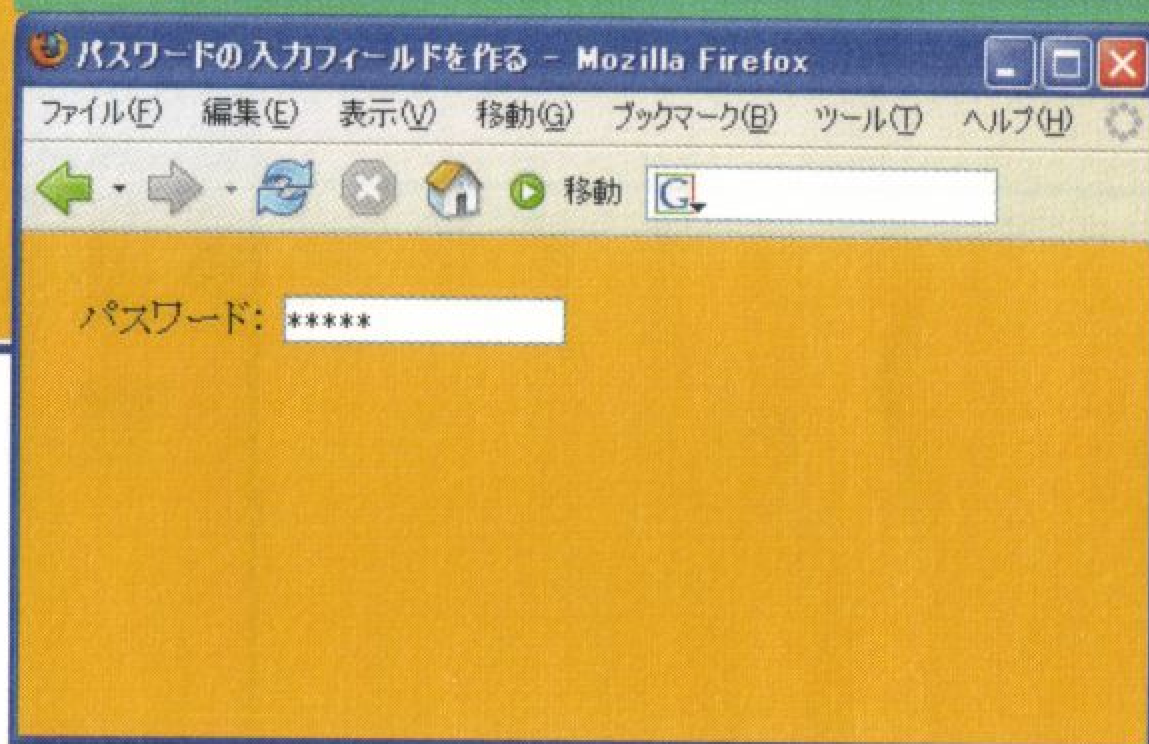
**<input type="password" name="*名前*" value="*デフォルト文字*" size="*幅*" maxlength="*最大文字数*">**

| | |
|---|---|
| 名前 | 入力フィールドの名前 |
| デフォルト文字 | あらかじめ入力されている文字 |
| 幅 | 入力フィールドの幅（文字数） |
| 最大文字数 | 入力可能な最大の文字数 |

Internet Explorer 6.0




Firefox 1.0




input要素の「type="password"」で、パスワードの入力に使用する1行のテキスト入力フィールドを作成します。
このフィールドに入力した文字は、他の文字や記号などに置き換えられて表示されます。name属性で指定する名前は、フォームの内容を受信した時にデータを見分けるためなどに使用されます。

## Sample

```
<p>
<label for="pw">パスワード：</label>
<input type="password" name="pword" size="20" id="pw">
</p>
```

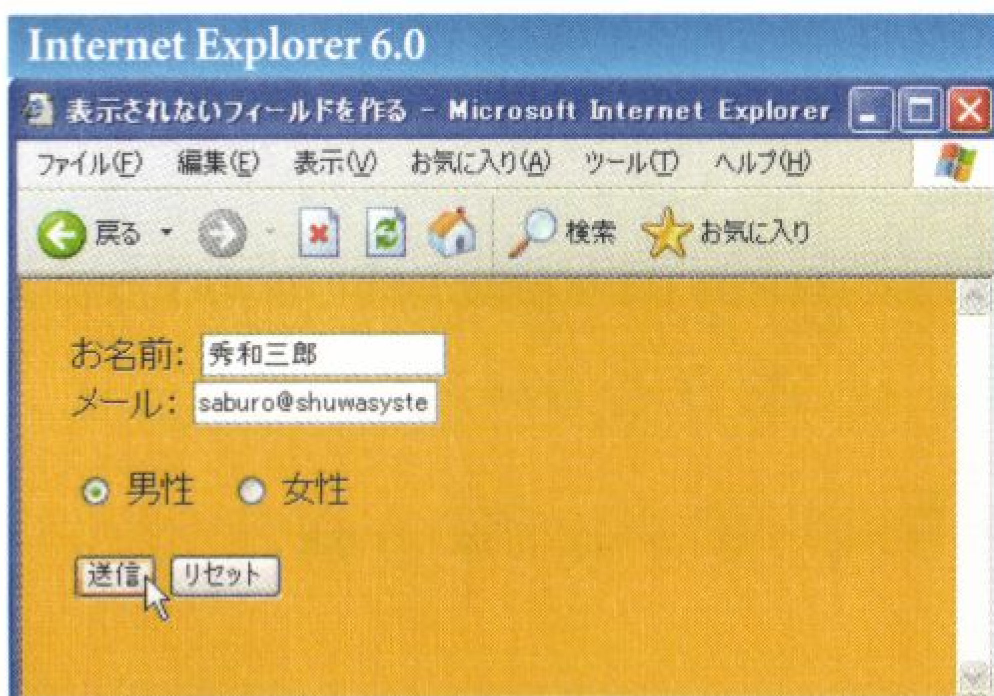
<label for="～">：「フォーム」の「ラベルテキストと項目を一体化させる」(P.132)

# 表示されないフィールドを作る

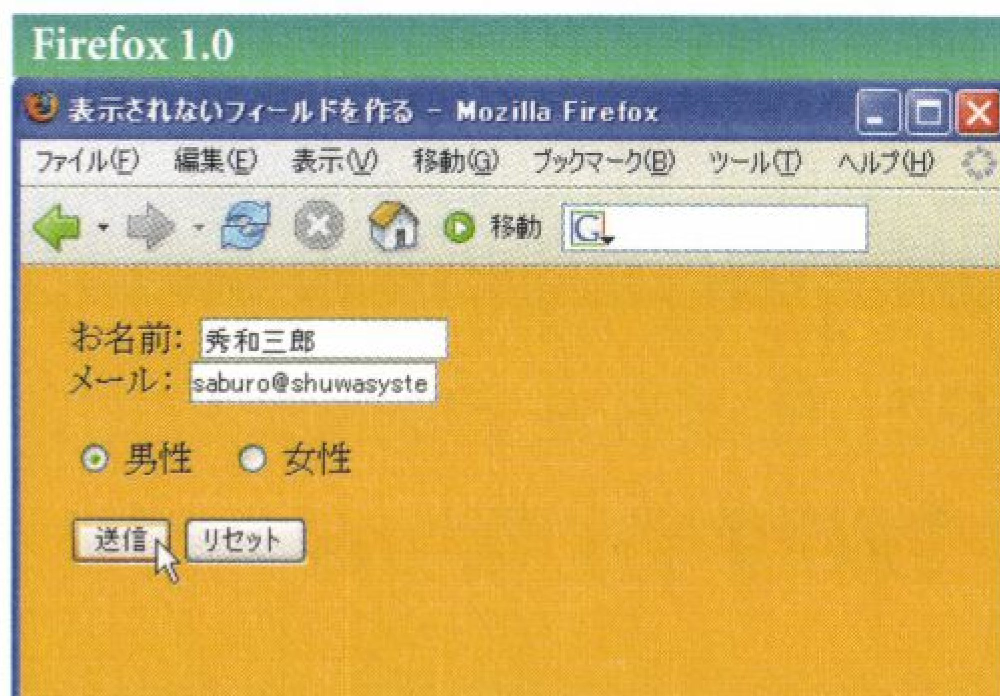
## <input type="hidden" name="名前" value="送信値">

| | |
|---|---|
| 名前 | フィールドの名前 |
| 送信値 | 送信する文字 |

Internet Explorer 6.0




Firefox 1.0




input要素の「type="hidden"」は、画面上には表示されないフィールドを作成します。一般に、ユーザーに見せる必要のない特定の値を、CGIプログラムに送信したい場合などに使用します。value属性で指定した内容が、固定値として送信されます。name属性で指定する名前は、フォームの内容を受信した側がこのデータを見分けるために使用されます。

### Sample

```
<form action="/cgi-bin/formmail.cgi" method="post">
<p>
<input type="hidden" name="recipient" value="hiroko@zspc.com">
<input type="hidden" name="subject" value="ユーザー登録">
<label for="nm">お名前：</label>
<input type="text" name="namae" id="nm"><br>
<label for="ma">メール：</label>
<input type="text" name="email" id="ma">
</p>
<p>
<input type="radio" name="sex" value="male" id="sm">
<label for="sm">男性</label>
<input type="radio" name="sex" value="female" id="sf">
<label for="sf">女性</label>
</p>
<p>
<input type="submit" value="送信">
<input type="reset" value="リセット">
</p>
</form>
```

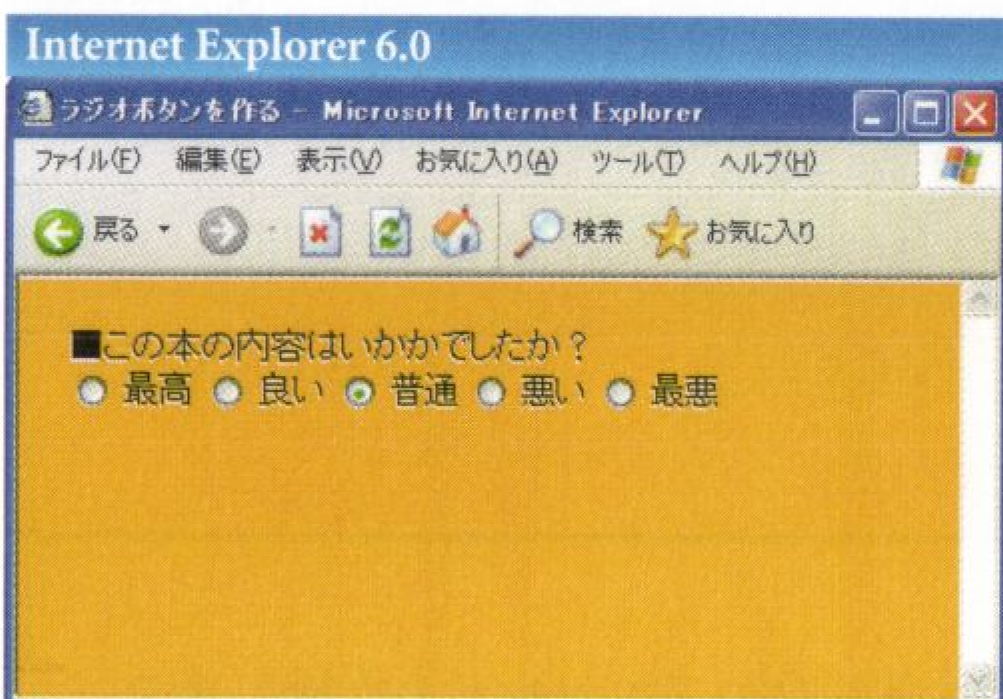
HTML

# ラジオボタンを作る

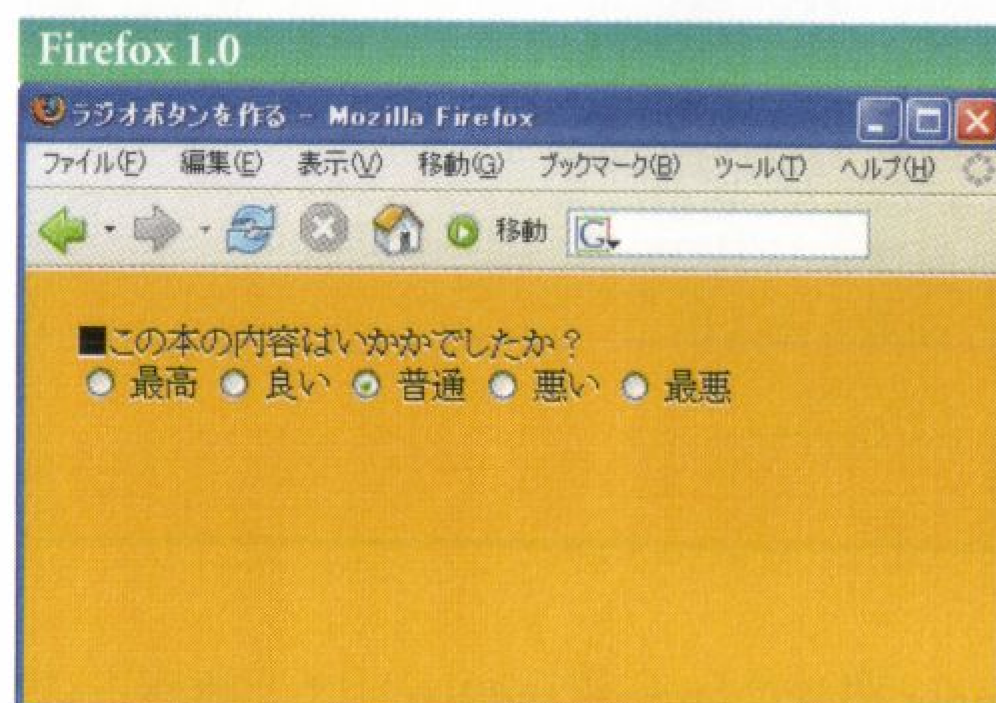
**<input type="radio" name="名前" value="送信文字">**
**<input type="radio" name="名前" value="送信文字" checked>**

| | |
|---|---|
| 名前 | ラジオボタンの名前 |
| 送信文字 | 選択された結果として送信される文字 |
| checked | あらかじめ選択された状態にする場合に指定 |

Internet Explorer 6.0




Firefox 1.0




input要素の「type="radio"」で、ラジオボタンを作成します。
ラジオボタンは、複数の選択項目のうちひとつだけ選択できる形式のボタンです。共通の項目に対する選択肢として使用するラジオボタンには、すべて同じ名前を指定する必要があります。また、データが送信された時にどの項目が選択されたのかを判別するために、value属性には個別の値を指定するようにしてください。

## Sample

```
<p>
●この本の内容はいかかでしたか？<br>
<input type="radio" name="book" value="best" id="v5">
<label for="v5">最高</label>
<input type="radio" name="book" value="good" id="v4">
<label for="v4">良い</label>
<input type="radio" name="book" value="normal" checked id="v3">
<label for="v3">普通</label>
<input type="radio" name="book" value="bad" id="v2">
<label for="v2">悪い</label>
<input type="radio" name="book" value="worst" id="v1">
<label for="v1">最悪</label>
</p>
```

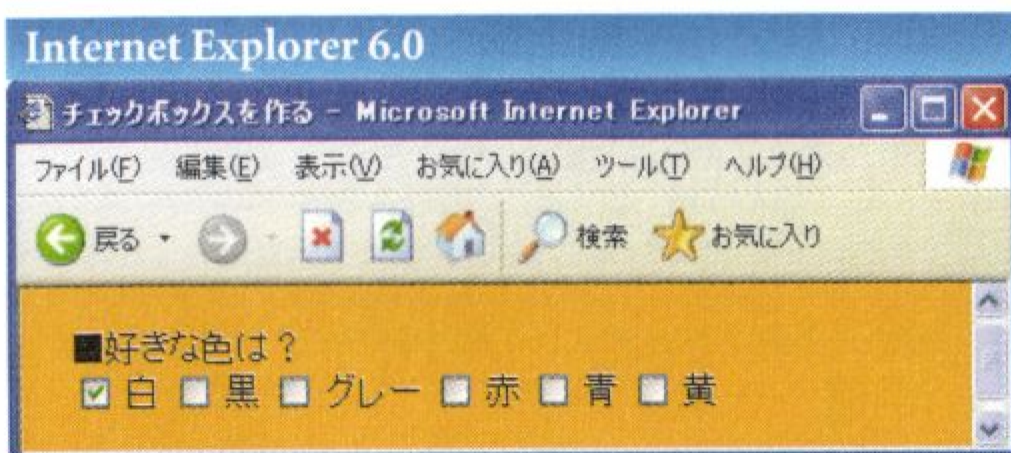
<label for="~">：「フォーム」の「ラベルテキストと項目を一体化させる」(P.132)

# チェックボックスを作る

**<input type="checkbox" name="名前" value="送信文字">**
**<input type="checkbox" name="名前" value="送信文字" *checked*>**

| | |
|---|---|
| 名前 | チェックボックスの名前 |
| 送信文字 | 選択された結果として送信される文字 |
| checked | あらかじめ選択された状態にする場合に指定 |

Internet Explorer 6.0




Firefox 1.0




input要素の「type="checkbox"」で、チェックボックスを作成します。
チェックボックスは、複数の選択項目の中から該当する項目を複数選択できるようにする場合に使用します。共通の項目に対する選択肢として使用するチェックボックスは、すべて同じ名前を指定する必要があります。また、データが送信された時にどの項目が選択されたのかを判別するために、value属性には個別の値を指定するようにしてください。

## Sample

```
<p>
●好きな色は？<br>
<input type="checkbox" name="color" value="white" checked id="c1">
<label for="c1">白</label>
<input type="checkbox" name="color" value="black" id="c2">
<label for="c2">黒</label>
<input type="checkbox" name="color" value="gray" id="c3">
<label for="c3">グレー</label>
<input type="checkbox" name="color" value="red" id="c4">
<label for="c4">赤</label>
<input type="checkbox" name="color" value="blue" id="c5">
<label for="c5">青</label>
<input type="checkbox" name="color" value="yellow" id="c6">
<label for="c6">黄</label>
</p>
```


<label for="～">：「フォーム」の「ラベルテキストと項目を一体化させる」(P.132)

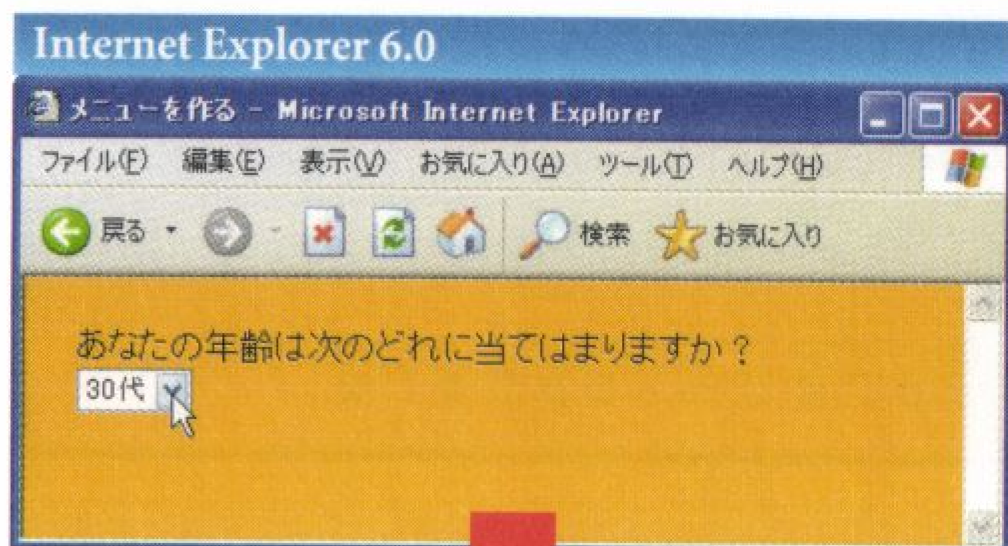
IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac i-mode

# メニューを作る

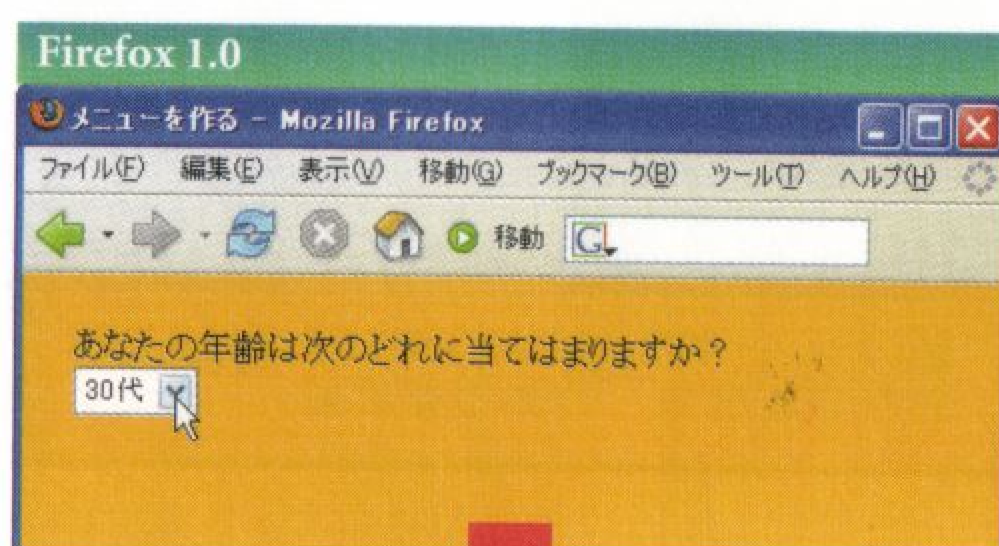
<select name="名前">～</select> ◆メニュー全体
<option value="送信値">～</option> ◆メニュー項目
<option selected>～</option> ◆メニュー項目

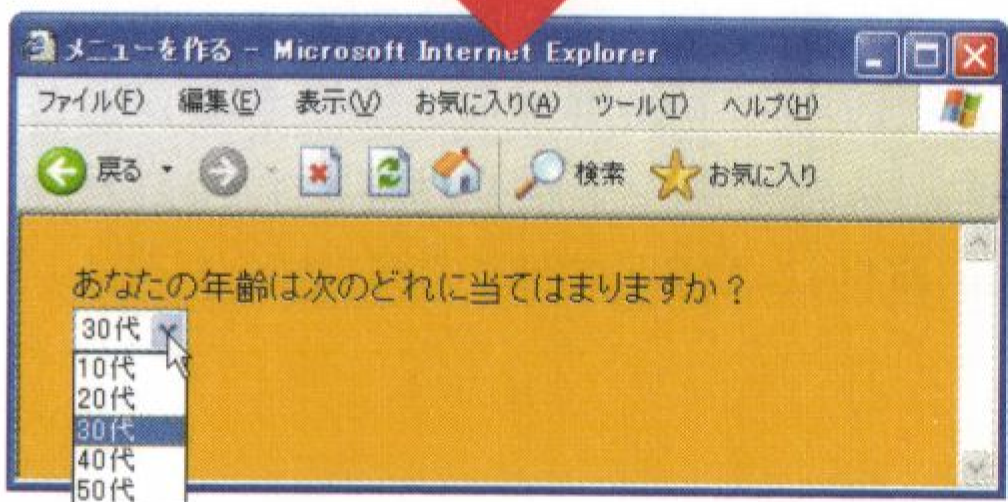
| | |
|---|---|
| 名前 | メニューの名前 |
| 送信値 | 選択された結果として送信される文字 |
| selected | あらかじめ選択された状態にする場合に指定 |

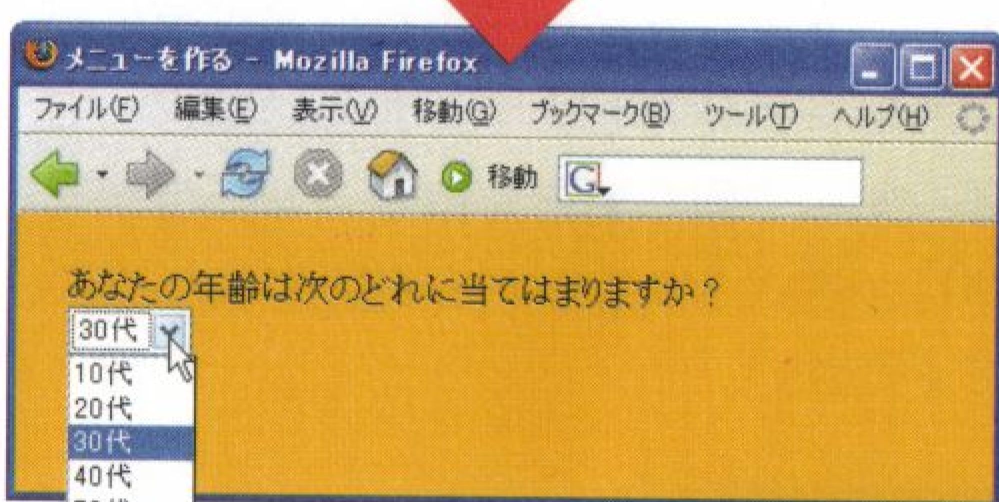
Internet Explorer 6.0




Firefox 1.0




select要素で、メニューを作成します。
メニュー全体を<select>～</select>で囲って示し、その中に選択肢を表す<option>～</option>を必要な数だけ配置します。<option>～</option>の範囲には、実際にメニューに表示される選択肢となるテキストを入れます。また、value属性を省略した場合には、ここに入れたテキスト自体が選択された項目として送信されます。

## Sample

```
<p>
あなたの年齢は次のどれに当てはまりますか？<br>
<select name="年齢">
  <option>10代</option>
  <option>20代</option>
  <option selected>30代</option>
  <option>40代</option>
  <option>50代</option>
</select>
</p>
```

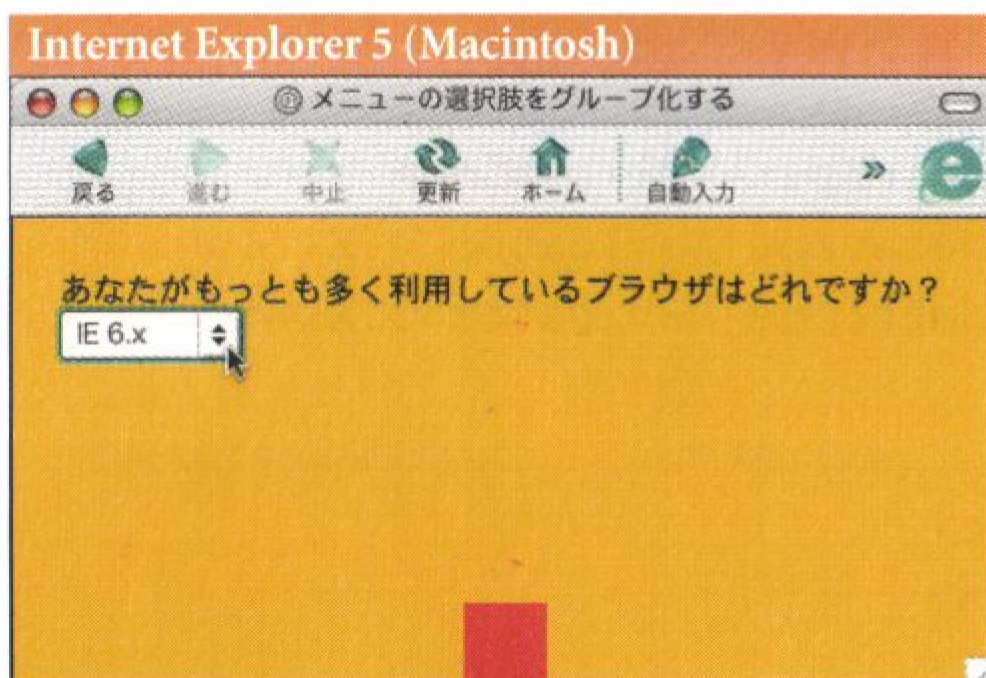
フォームを作る

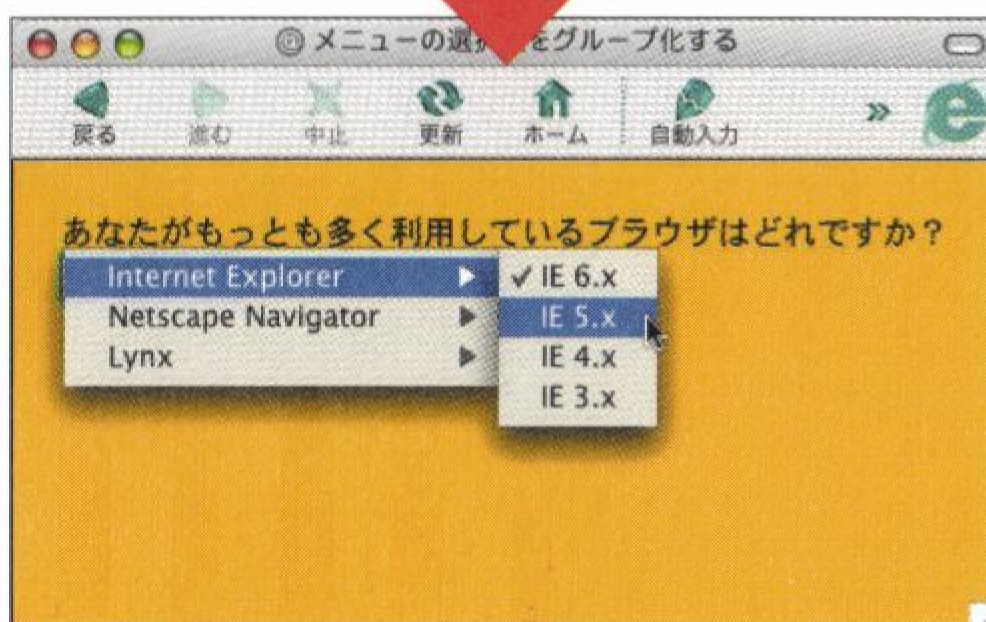
# メニューの選択肢をグループ化する

**<optgroup label="*グループ名*">～</optgroup>** ⬅グループを作成
**<option label="*短い選択肢*">～</option>** ⬅グループ内の項目

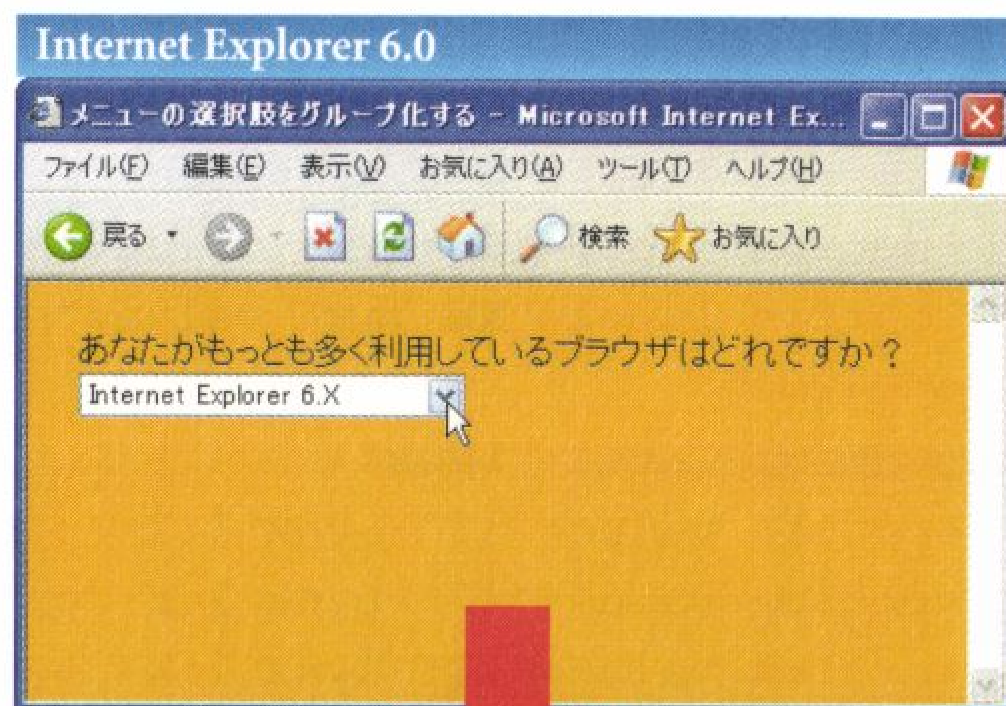
| | |
|---|---|
| グループ名 | 第1階層で表示されるグループのタイトル |
| 短い選択肢 | グループ名に対応させた第2階層で表示される短い選択肢 |

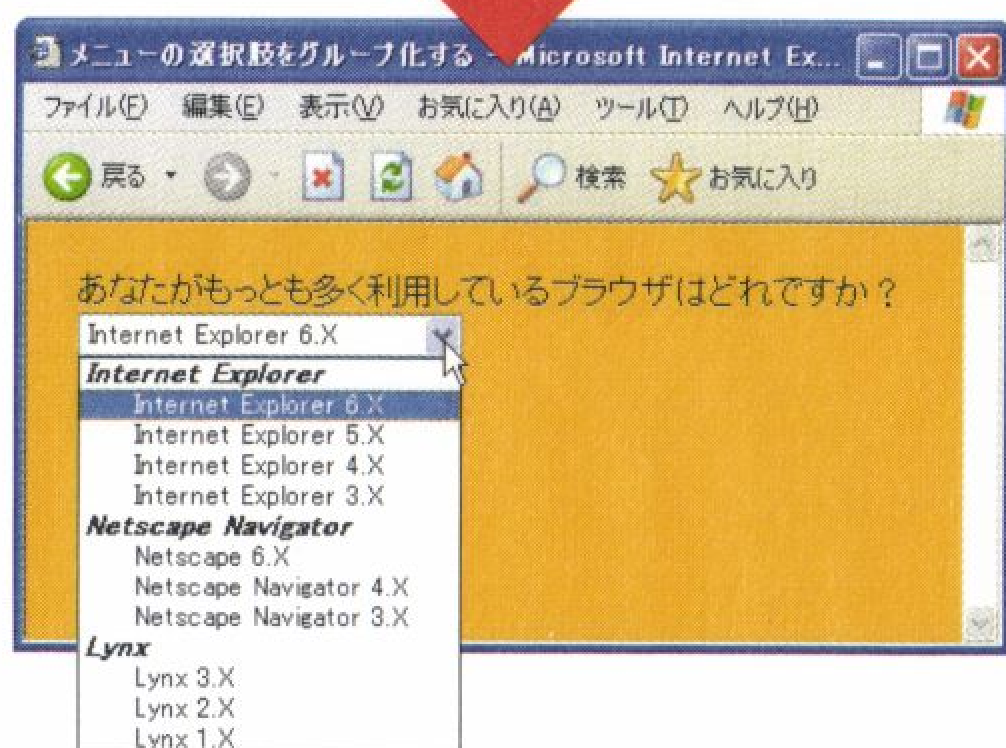
Internet Explorer 5 (Macintosh)




Internet Explorer 6.0




select要素で作成されるメニューの選択肢をグループ化します（結果としてメニューは階層化されます）。

optgroup要素のlabel属性の値は、グループ（第1階層）のタイトルとしてメニューに表示されるものですので、必ず指定するようにしてください。option要素のlabel属性の値には、グループ名が表示されることによって省略できる部分を省いた短い選択肢を指定します。この属性を省略した場合には、<option>～</option>の範囲に指定されている内容がそのまま利用されます。

## Sample

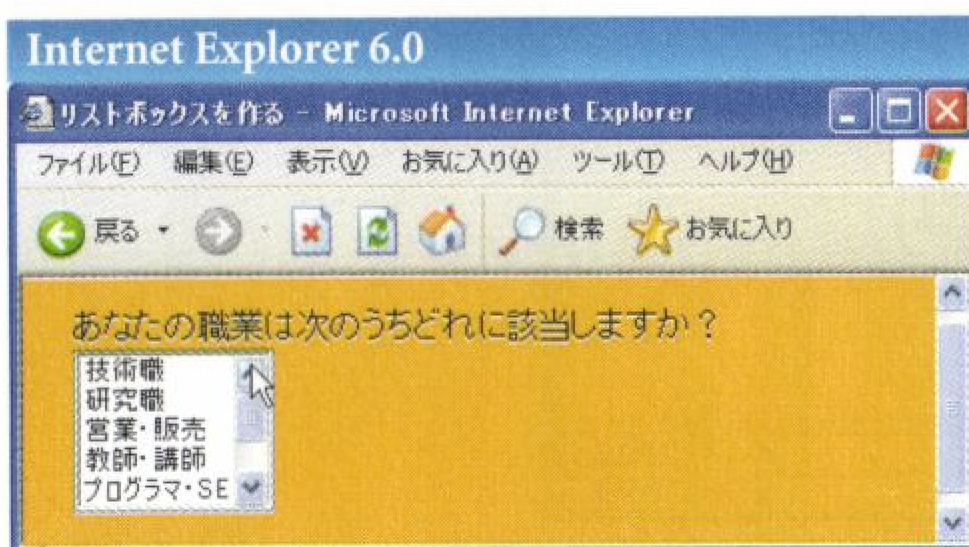
```
<p>
あなたがもっとも多く利用しているブラウザはどれですか？<br>
<select name="browser">
<optgroup label="Internet Explorer">
  <option label="IE 6.x" value="e6">Internet Explorer 6.X</option>
  <option label="IE 5.x" value="e5">Internet Explorer 5.X</option>
  <option label="IE 4.x" value="e4">Internet Explorer 4.X</option>
  <option label="IE 3.x" value="e3">Internet Explorer 3.X</option>
</optgroup>
<optgroup label="Netscape Navigator">
  <option label="N 6.x" value="n6">Netscape 6.X</option>
  <option label="NN 4.x" value="n4">Netscape Navigator 4.X</option>
  <option label="NN 3.x" value="n3">Netscape Navigator 3.X</option>
</optgroup>
<optgroup label="Lynx">
  <option value="l3">Lynx 3.X</option>
  <option value="l2">Lynx 2.X</option>
  <option value="l1">Lynx 1.X</option>
</optgroup>
</select>
</p>
```

# リストボックスを作る

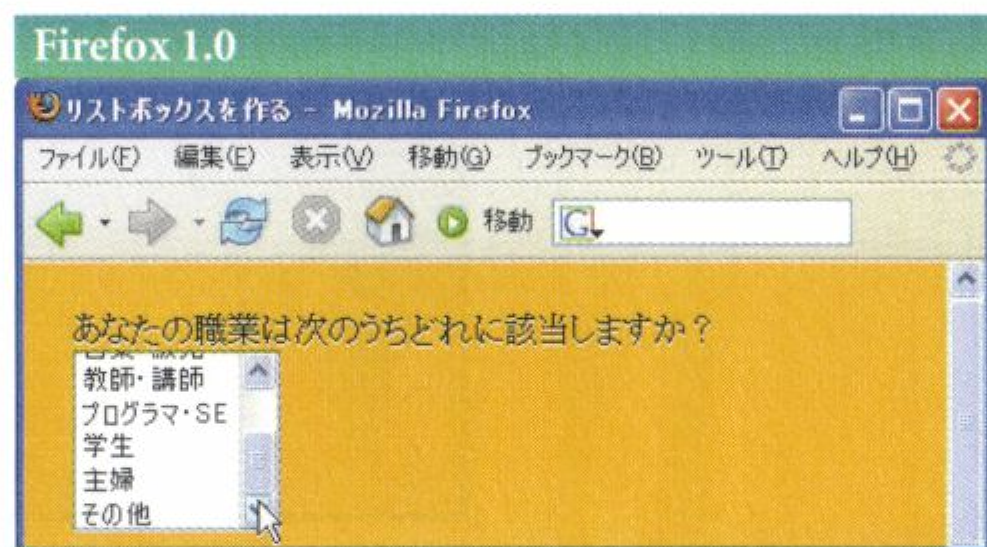
<select size="*行数*" name="*名前*" multiple>～</select> ◆リストボックス
<option value="*送信値*">～</option> ◆選択項目
<option *selected*>～</option> ◆選択項目

| | |
|---|---|
| 行数 | リストボックスの表示行数 |
| 名前 | リストボックスの名前 |
| multiple | 複数の項目を選択可能にする場合に指定 |
| 送信値 | 選択された結果として送信される文字 |
| selected | あらかじめ選択された状態にする場合に指定 |

Internet Explorer 6.0




Firefox 1.0




メニューを作成するselect要素にsize属性を指定すると、リストボックスとして表示されます。
メニューの場合と同様に、リストボックス全体を<select>～</select>で囲って示し、その中に選択肢を表す<option>～</option>を必要な数だけ配置します。<option>～</option>の範囲には、実際にリストボックスに表示される選択肢のテキストを入れます。また、value属性を省略した場合は、ここに入れたテキスト自体が選択された項目として送信されます。option属性で示される選択肢の数がsize属性で示される表示行数より多い場合には、リストボックスに自動的にスクロールバーが付けられます。

## Sample

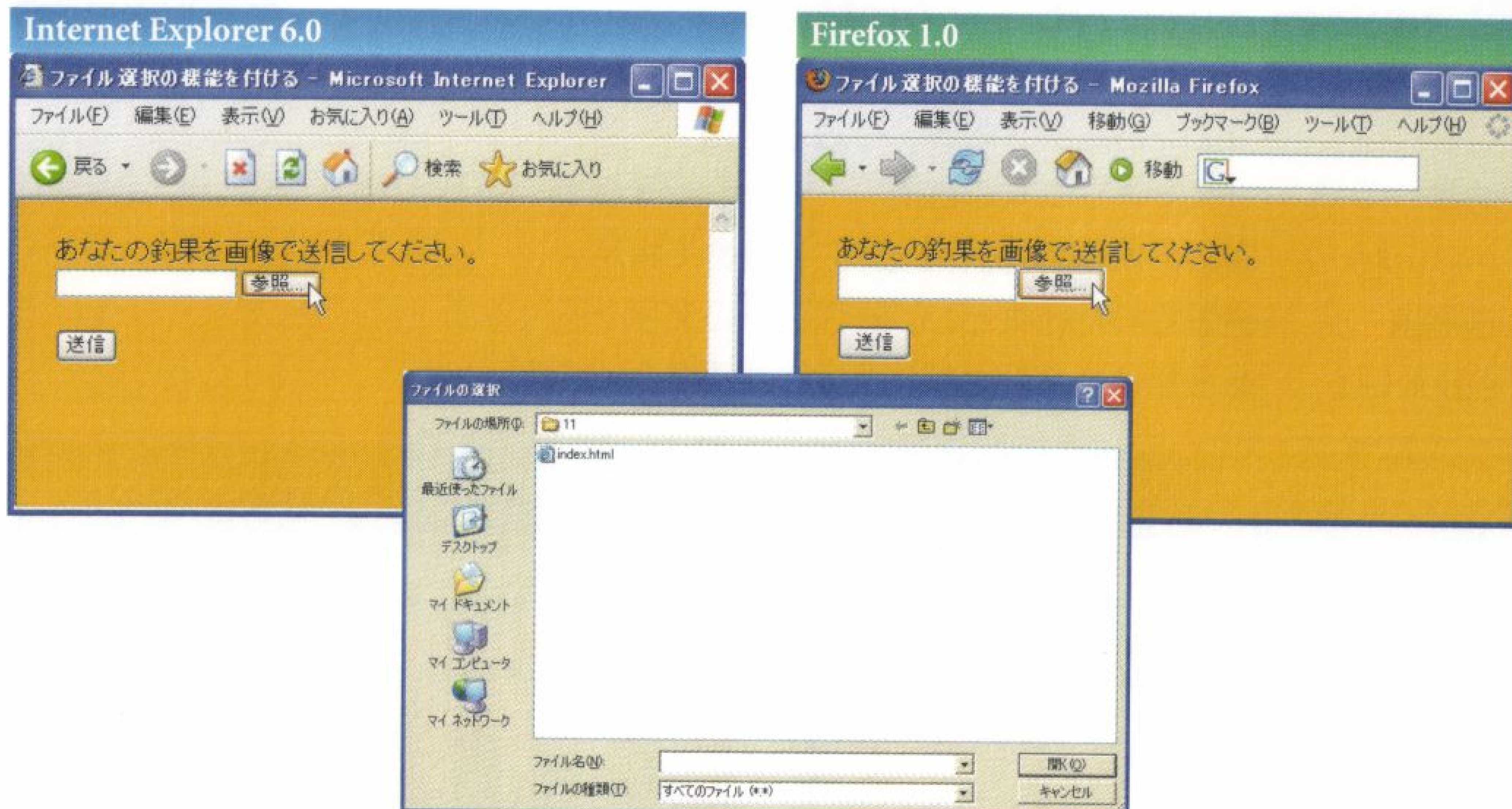
```
<p>
あなたの職業は次のうちどれに該当しますか？<br>
<select size="5" name="occupation" multiple>
  <option>技術職</option>
  <option>研究職</option>
  <option>営業・販売</option>
  <option>教師・講師</option>
  <option>プログラマ・SE</option>
  <option>学生</option>
  <option>主婦</option>
  <option>その他</option>
</select>
</p>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# ファイル選択の機能を付ける

**<input type="file" name="名前" accept="*MIME*タイプ">**

| | |
|---|---|
| MIMEタイプ | 受け付け可能な「,」区切りのMIMEタイプリスト |




input要素の「type="file"」は、フォームのデータとして送信するファイルを選択できるようなボタンとフィールドを自動的に作成します。
accept属性には、受信プログラムが受け付けることのできるファイルの種類をMIMEタイプで指定します。複数の種類を受信可能な場合には、それらを「,」で区切って指定することができます。
ただし、現状では、この属性はほとんどサポートされていないようです。

※この機能を利用する場合、form要素のmethod属性には「post」を、enctype属性には「multipart/form-data」を指定する必要があります。

## Sample

```
<form action="/cgi-bin/snap.cgi" enctype="multipart/form-data" 
 method="post">
<p>
あなたの釣果を画像で送信してください。<br>
<input type="file" name="imagefile" accept="image/jpeg,image/gif">
</p>
<p>
<input type="submit" value="送信">
</p>
</form>
```

フォームを作る

# 項目をグループ化する

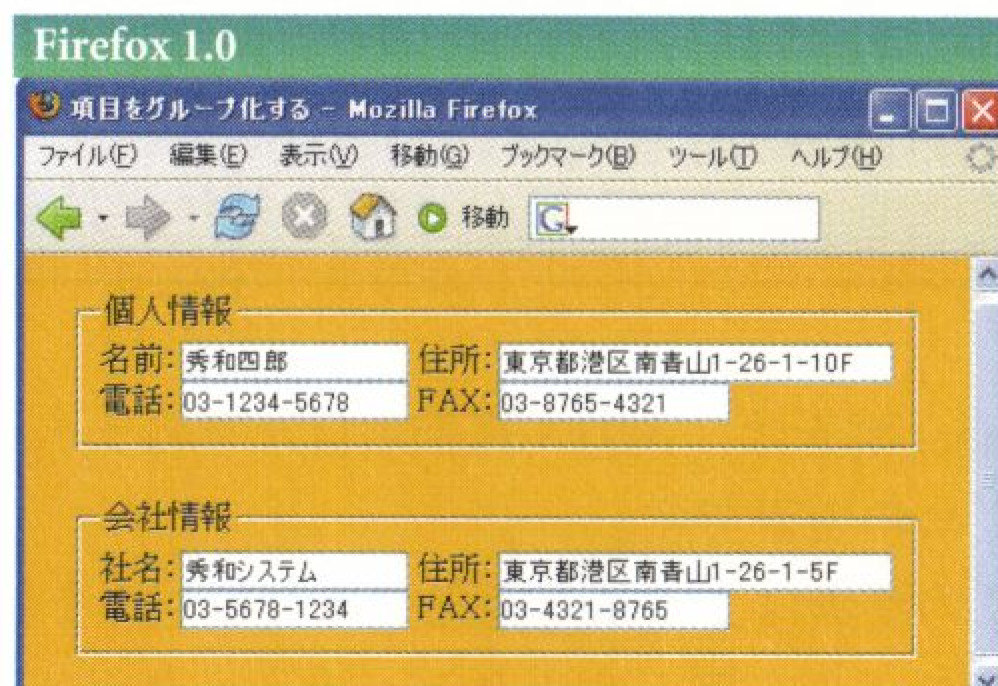
<fieldset>～</fieldset> ←グループ化
<legend align="位置">～</legend> ←グループのタイトル

| 【位置】 | |
|---|---|
| top | タイトルを上に表示する（デフォルト） |
| bottom | タイトルを下に表示する |
| left | タイトルを左に表示する |
| right | タイトルを右に表示する |

Internet Explorer 6.0




Firefox 1.0




fieldset要素は、フォームに含まれる入力項目や選択項目をグループ化します。<fieldset>～</fieldset>の範囲の先頭にはlegend要素を配置して、そのグループのタイトルを付けます。align属性は非推奨とされていますが、現在のところ、これに代わるスタイルシートのプロパティは定義されていません。

## Sample

```
<fieldset>
<legend>個人情報</legend>
<p>
名前：<input type="text" name="pname">
住所：<input type="text" name="paddrs" size="36">
<br>
電話：<input type="text" name="pphone">
FAX：<input type="text" name="pfax">
</p>
</fieldset>
<fieldset>
<legend>会社情報</legend>
<p>
社名：<input type="text" name="cname">
住所：<input type="text" name="caddrs" size="36">
<br>
電話：<input type="text" name="cphone">
FAX：<input type="text" name="cfax">
</p>
</fieldset>
```

# ラベルテキストと項目を一体化させる

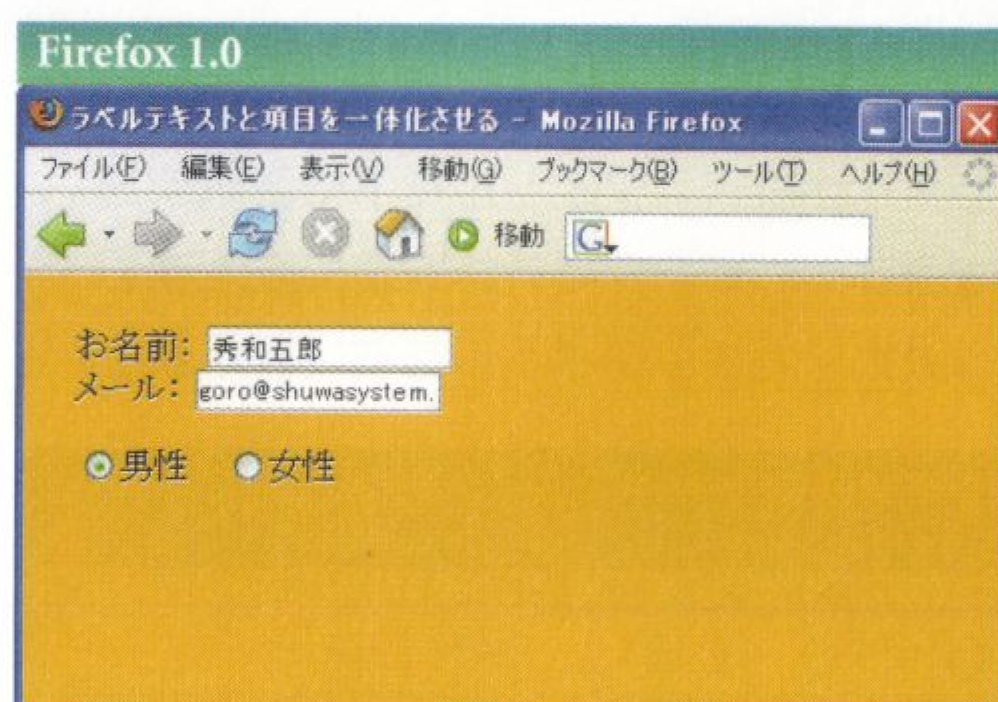
## <label for="参照ID">～</label>

| | |
|---|---|
| 参照ID | ラベルを付ける対象のid属性の値 |

Internet Explorer 6.0




Firefox 1.0




label要素は、入力・選択項目とそのラベルテキストを明確に関連付けて一体化させるための要素です。
この要素は、value属性によってラベルを付けることのできない項目(入力フィールド・メニュー・ラジオボタン・チェックボックスなど)に対して使用します。これによって、たとえばラジオボタンやチェックボックスは、ラベルとして付けられたテキスト部分をクリックしても反応して切り替わるようになります。
ラベルの付け方には、2通りの方法があります。ひとつは、<label>～</label>の範囲内にラベルとなるテキストと入力・選択項目の両方を含める方法です。もうひとつは、<label>～</label>の範囲内にはラベルとなるテキストのみを配置して、入力・選択項目のid属性で指定した値と同じものをfor属性で指定する方法です。この場合は、ラベルと入力・選択項目が必ず1対1になるようにしてください。現在のところ、Internet Explorerはfor属性を使用する方法にしか対応していないようです。

### Sample

```
<p>
<label for="nm">お名前：</label>
<input type="text" name="namae" id="nm"><br>
<label for="em">メール：</label>
<input type="text" name="email" id="em">
</p>
<p>
<label><input type="radio" name="sex" value="Male">男性</label>
<label><input type="radio" name="sex" value="Female">女性</label>
</p>
```

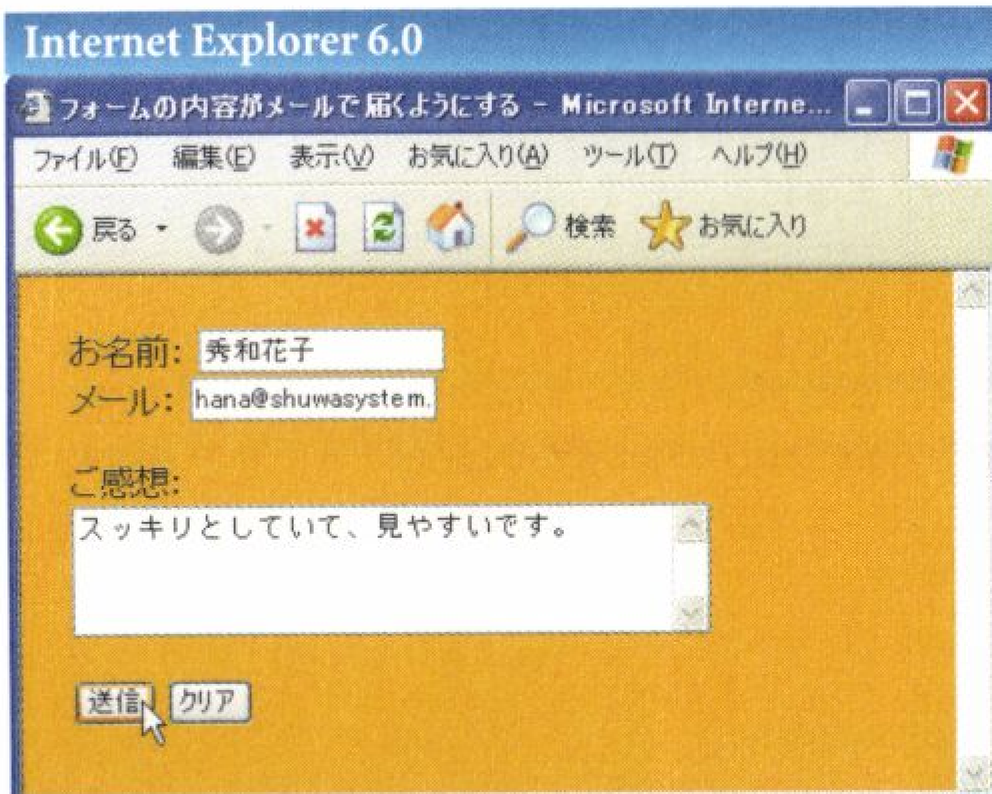
TIPS「ラベルについて」(P.134)

# フォームの内容がメールで届くようにする

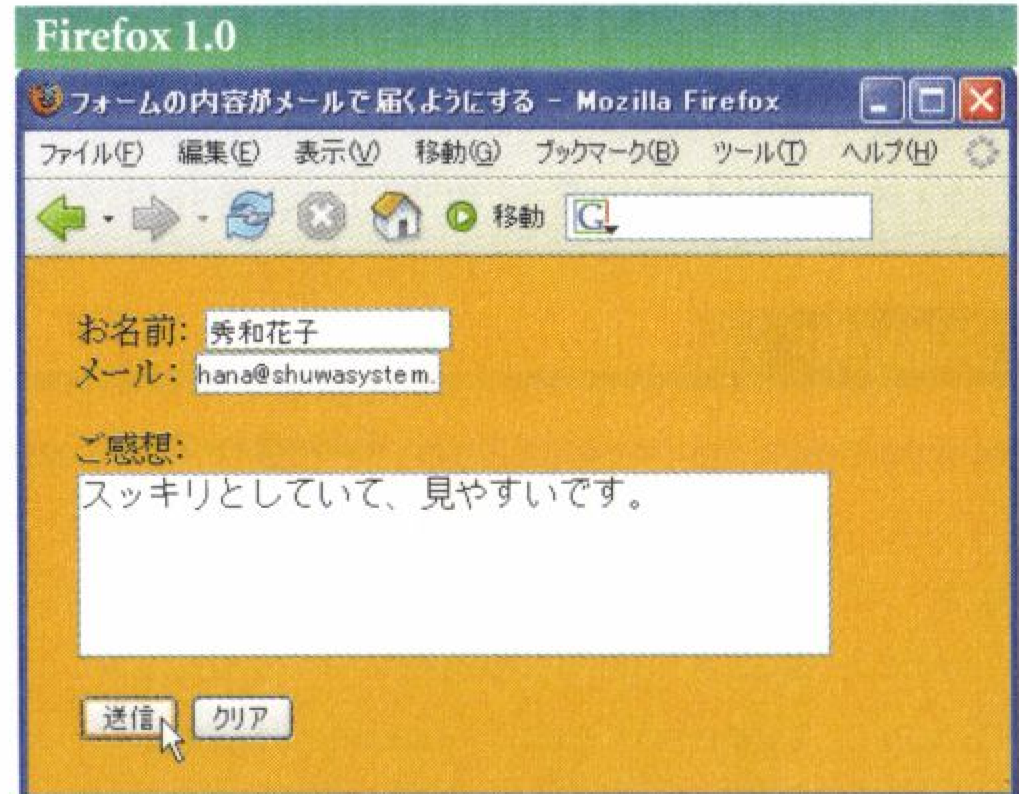
**<form action="mailto:*メールアドレス*" method="post" enctype="*MIMEタイプ*">～</form>**

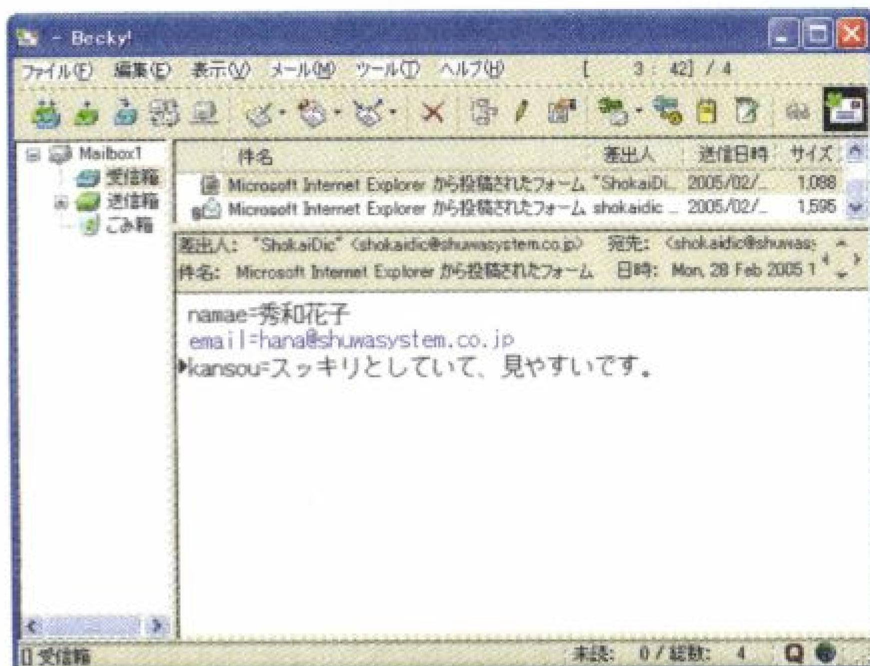
| | |
|---|---|
| メールアドレス | フォームの送信先メールアドレス |
| MIMEタイプ | メールで送信する際のMIMEタイプ |

Internet Explorer 6.0

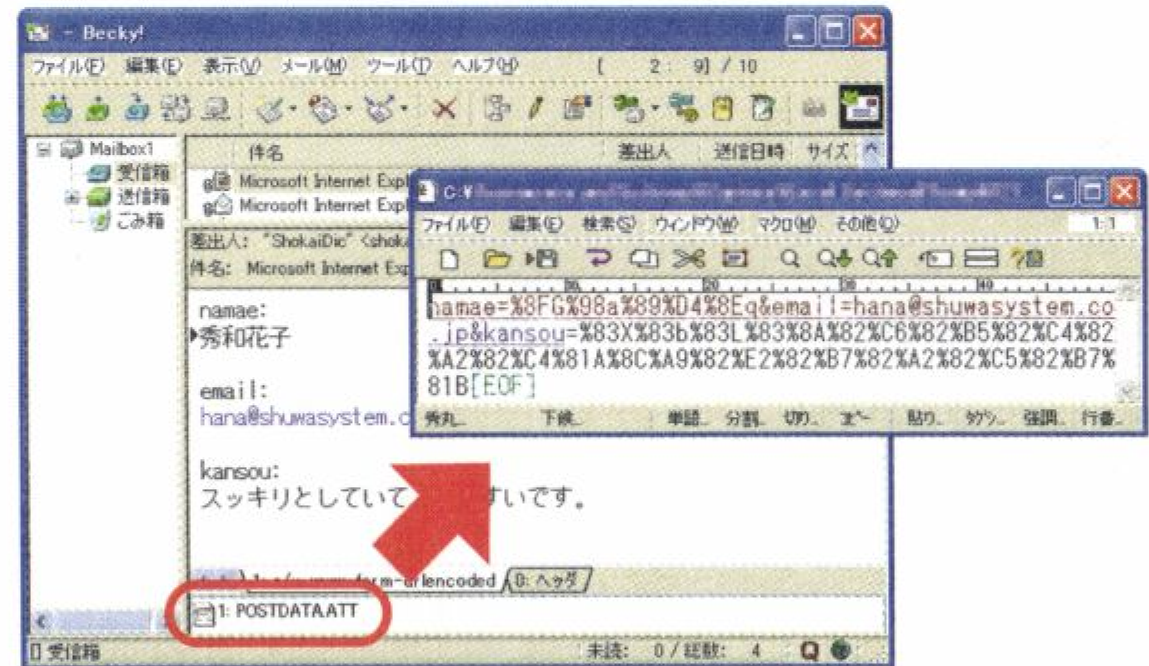



Firefox 1.0

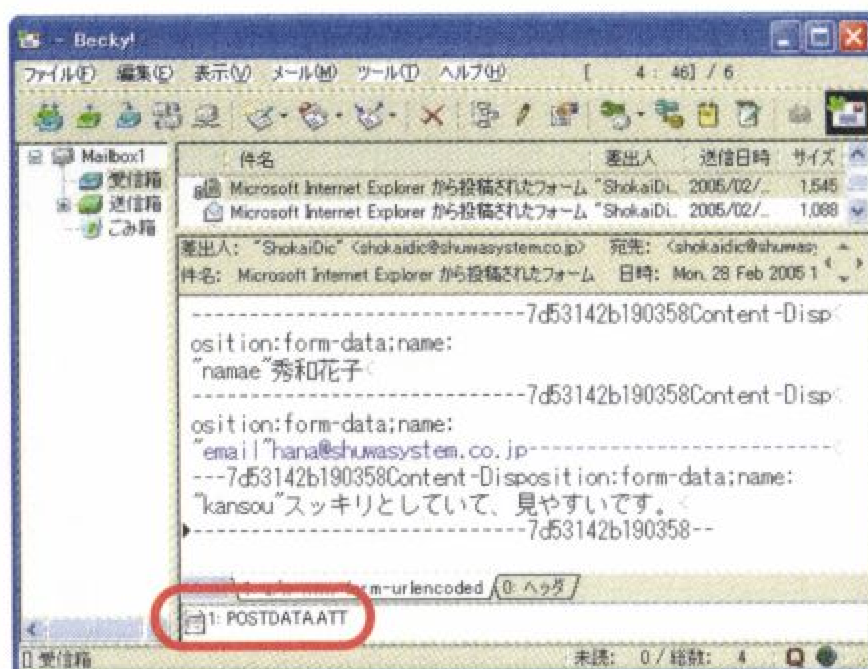


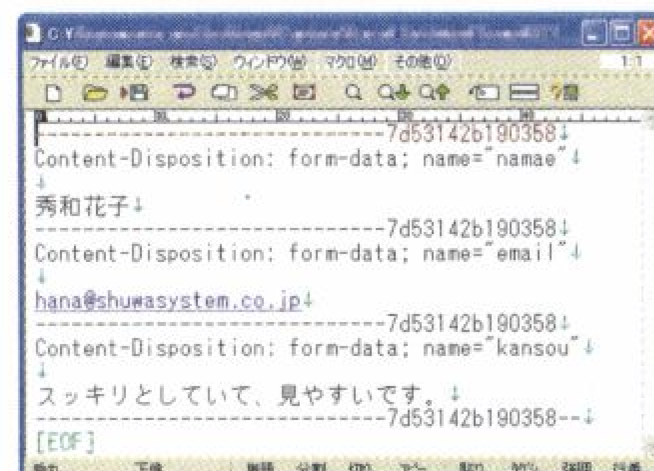




enctype="text/plain"の場合の受信例




enctype="application/x-www-form-urlencoded"の場合の受信例




enctype="multipart/form-data"の場合の受信例

フォームの内容は、メールとして送信することもできます。その場合は、action属性に「mailto:メールアドレス」を指定します。

enctype属性を指定しない場合には、デフォルトの値としてMIMEタイプに「application/x-www-form-urlencoded」が採用されます。この場合は、送られてくるメールはそのままでは内容を読むことができない状態になっていますので、それを変換するためのソフトウェアが必要になります。enctype属性に「text/plain」または「multipart/form-data」を指定すると、メーラーでそのまま読める状態で送信させることができます。ただし、フォームのデータを受信するメーラーの種類やその設定、送信するブラウザがフォームの内容をメールとして送信する機能を備えているかどうかによっては、うまく機能しない場合があります。確実にデータを送信して欲しいのであれば、メールで送信する形式にはせずにCGIを利用してください。

実際に試す場合には、サンプルの「****@******.ne.jp」の部分を、メールを受け取りたいメールアドレスに変更してから実行してください。

## Sample

```
<form action="mailto:****@******.ne.jp" method="post" 
enctype="text/plain">
<p>
<label for="nm">お名前：</label>
<input type="text" name="namae" id="nm"><br>
<label for="ad">メール：</label>
<input type="text" name="email" id="ad">
</p>
<p>
ご感想：<br>
<textarea name="kansou" rows="4" cols="40">
ここにご感想をどうぞ。
</textarea>
</p>
<p>
<input type="submit" value="送信">
<input type="reset" value="クリア">
</p>
</form>
```

### TIPS

**ラベルについて**

一般のアプリケーションを使用している場合、ラジオボタンやチェックボックスは、ボタン部分ではなくテキストの部分をクリックしても反応します。しかし、label要素が登場するまでは、Webページ上のラジオボタンやチェックボックスは、そのようには動作しませんでした。つまり、ボタン部分とテキスト部分の関連を明確に示す方法がなかったため、それらがひとつのものとして認識されず、ボタンそのものをクリックしなければ反応しない状態になっていたのです。

label要素をサポートしているブラウザはまだ多いとはいえませんが、この要素を指定したからといって未対応のブラウザで特に問題が発生するわけではありません。上記のような不自然な動きをさせないためにも、label要素は常に指定するようにしましょう。

# フレームの全体構造を指定する

**<frameset rows="*高さ*">～</frameset>**
**<frameset cols="*幅*">～</frameset>**
**<frame src="*URL*" name="*フレーム名*">**

| | |
|---|---|
| 高さ | 横に分割する個数分の高さを上から順に「,」区切りで指定(ピクセル・%・*) |
| 幅 | 縦に分割する個数分の幅を左から順に「,」区切りで指定(ピクセル・%・*) |
| URL | フレームの内容として表示するHTML文書のURL |
| フレーム名 | リンクなどの表示先として指定する場合に利用する名前 |





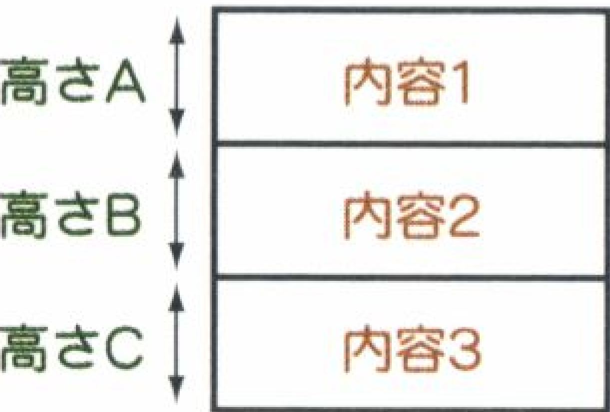


フレーム機能を利用すると、ウィンドウを縦横に区切って、その中にそれぞれ別のHTML文書を表示させることができます。
ウィンドウをどのように区切るかを指定するのがframeset要素で、分割された各フレームに表示する内容(HTML文書)を指定するのがframe要素です。フレームを指定する文書では、本来body要素があるべき部分に、代わりにframeset要素を配置します。body要素は使用できませんので、注意してください(ただし、後述する<noframes>～</noframes>の中には必要です)。

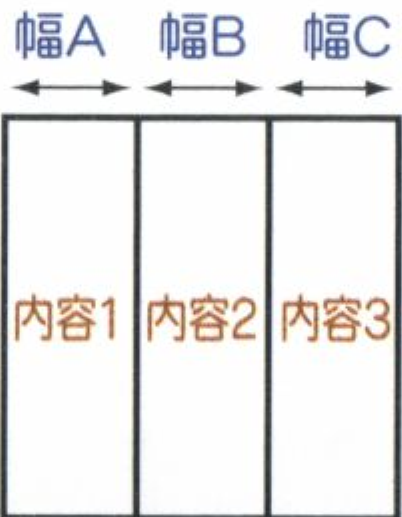
frameset要素のrows属性は横方向に分割する場合のフレームの各高さを上から順に、cols属性は縦方向に分割する場合のフレームの各幅を左から順に、「,」で区切って指定します。<frameset>～</frameset>の範囲には、分割される個数分の内容を順に入れる必要があります。フレームをそれ以上分割しないのであればframe要素で読み込むHTML文書を指定し、さらに分割する場合にはframeset要素を配置して(入れ子にして)そのフレームの分割を指定します。

※フレームを定義するHTML文書では、<!DOCTYPE>として「Frameset」を指定することに注意してください。

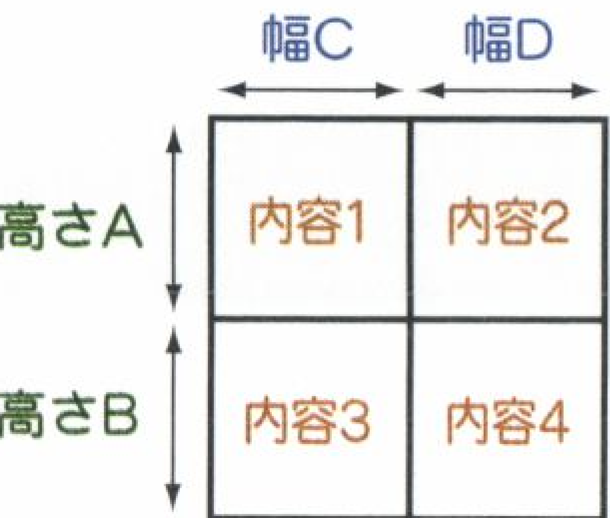
**基本的な分割の指定例**



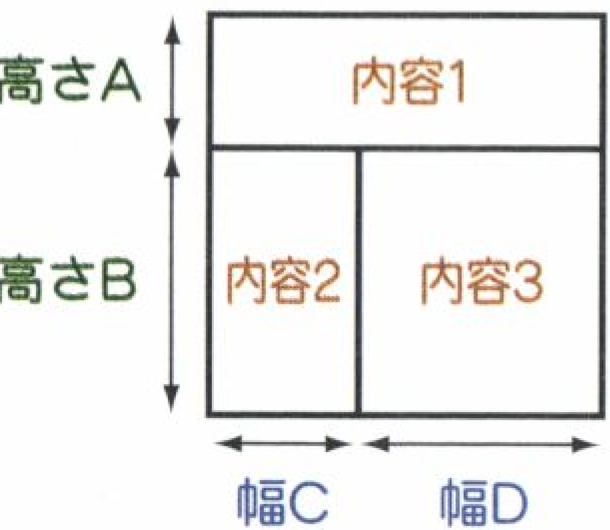

```
<frameset rows="高さA,高さB,高さC">
  <frame src=内容1.html">
  <frame src=内容2.html">
  <frame src=内容3.html">
</frameset>
```




```
<frameset cols="幅A,幅B,幅C">
  <frame src=内容1.html">
  <frame src=内容2.html">
  <frame src=内容3.html">
</frameset>
```




```
<frameset rows="高さA,高さB" cols="幅C,幅D">
  <frame src=内容1.html">
  <frame src=内容2.html">
  <frame src=内容3.html">
  <frame src=内容4.html">
</frameset>
```




```
<frameset rows="高さA,高さB">
  <frame src=内容1.html">
  <frameset cols="幅C,幅D">
    <frame src=内容2.html">
    <frame src=内容3.html">
  </frameset>
</frameset>
```

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
 "http://www.w3.org/TR/html4/frameset.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>フレームの全体構成サンプル</title>
</head>
<frameset rows="70,*">
  <frame src="title.html" name="logo">
  <frameset cols="150,*">
    <frame src="menu.html" name="menu">
    <frame src="content.html" name="content">
  </frameset>
  <noframes>
  <body>
   ∫
  </body>
  </noframes>
</frameset>
</html>
```


コラム「HTML4.01の種類と<!DOCTYPE～>の書き方」(P.10)
<noframes>：「フレーム」の「フレームが表示されない環境用の内容を入れる」(P.143)

**TIPS**

## 「*」による幅や高さの割合の指定

frameset要素のrows属性とcols属性や、col要素とcolgroup要素のwidth属性で長さ（幅・高さ）を複数指定する場合、ピクセルや%だけでなく、比率を指定することもできます。その場合に使用するのが「*」記号で、「*」の前に整数を付けて表します。この場合、まずピクセルや%で指定されている長さがあればその長さが確保され、残りの部分が指定された比率で分配されることになります。たとえば、「40,1*,2*,3*」と指定された場合は、40ピクセルを除いた残りの長さを「1:2:3」の割合で分割します。

なお、単に「*」だけが指定された場合は、「1*」と同じ意味になります。

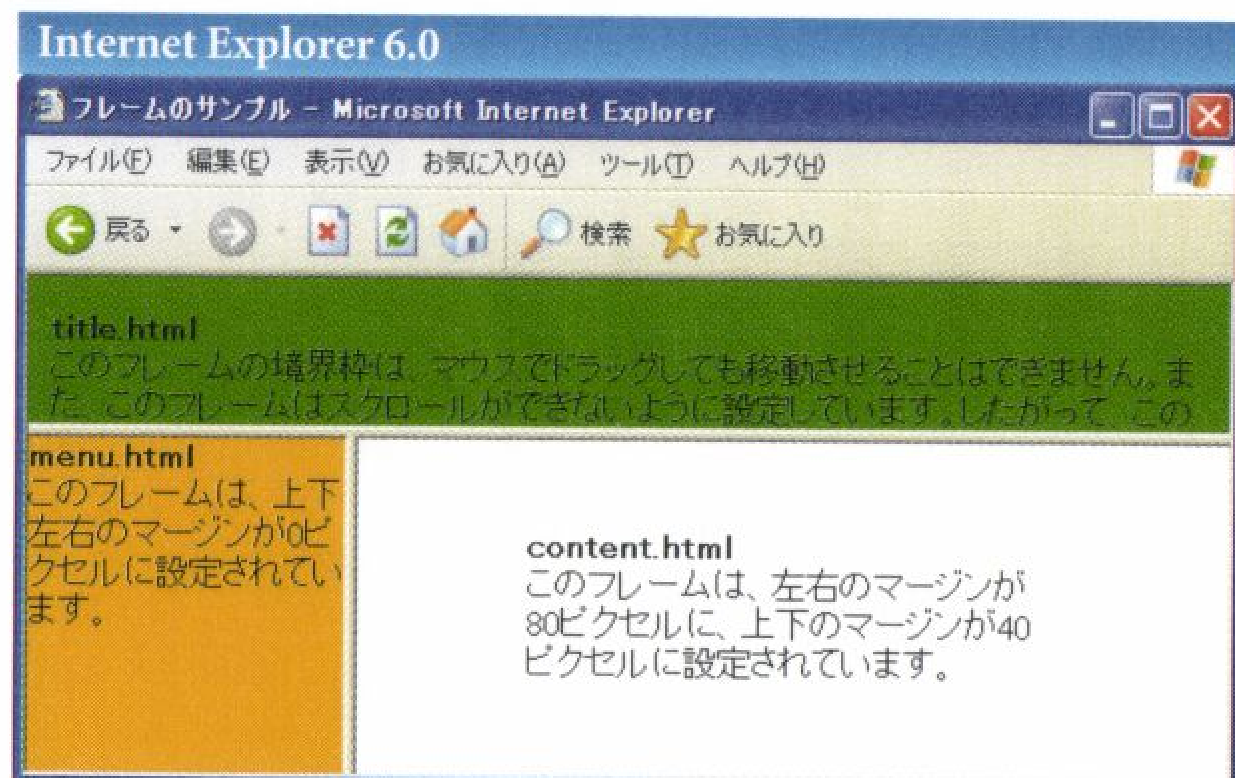
IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# フレームの表示方法を設定する

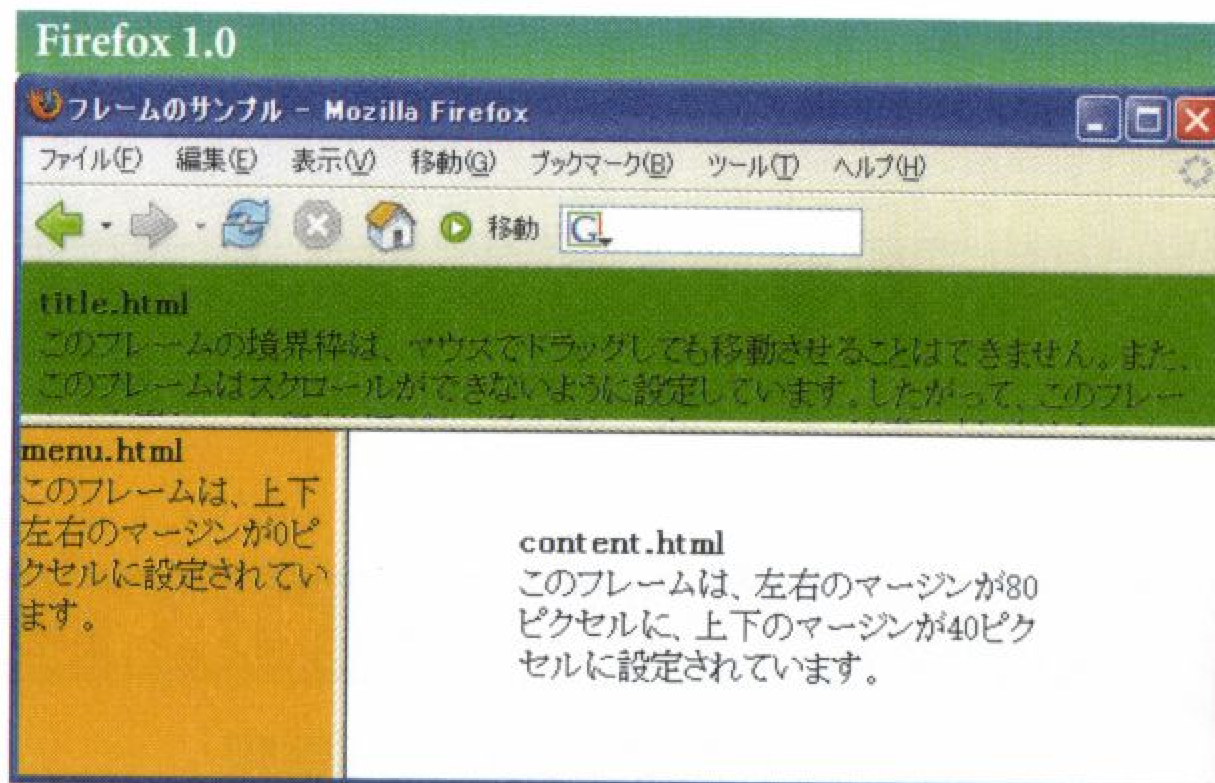
**<frame scrolling="スクロールの制御" noresize>**
**<frame marginwidth="左右のマージン" marginheight="上下のマージン">**

| | |
|---|---|
| noresize | フレームのサイズ変更を禁止 |
| 左右のマージン | フレーム内の左右のマージン(ピクセル) |
| 上下のマージン | フレーム内の上下のマージン(ピクセル) |
| 【スクロールの制御】 | |
| auto | 必要に応じてスクロール可能(デフォルト) |
| yes | 常にスクロール可能(スクロールバーを表示) |
| no | 常にスクロール不可(スクロールバーを非表示) |

Internet Explorer 6.0




Firefox 1.0




scrolling属性は、そのフレームの内容を表示する場合にスクロールができるようにするかどうかを設定をします。
noresize属性は、各フレームを区切る境界の枠をマウスで移動できないようにして、フレームの表示領域を固定します。marginwidth属性とmarginheight属性は、そのフレーム内でのマージンを設定します。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
 "http://www.w3.org/TR/html4/frameset.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>フレームのサンプル</title>
</head>
<frameset rows="70,*">
  <frame src="title.html" scrolling="no" noresize>
  <frameset cols="150,*">
    <frame src="menu.html" marginwidth="0" marginheight="0">
    <frame src="content.html" marginwidth="80" marginheight="40">
  </frameset>
  <noframes>
  <body>
   ⁓
  </body>
  </noframes>
</frameset>
</html>
```

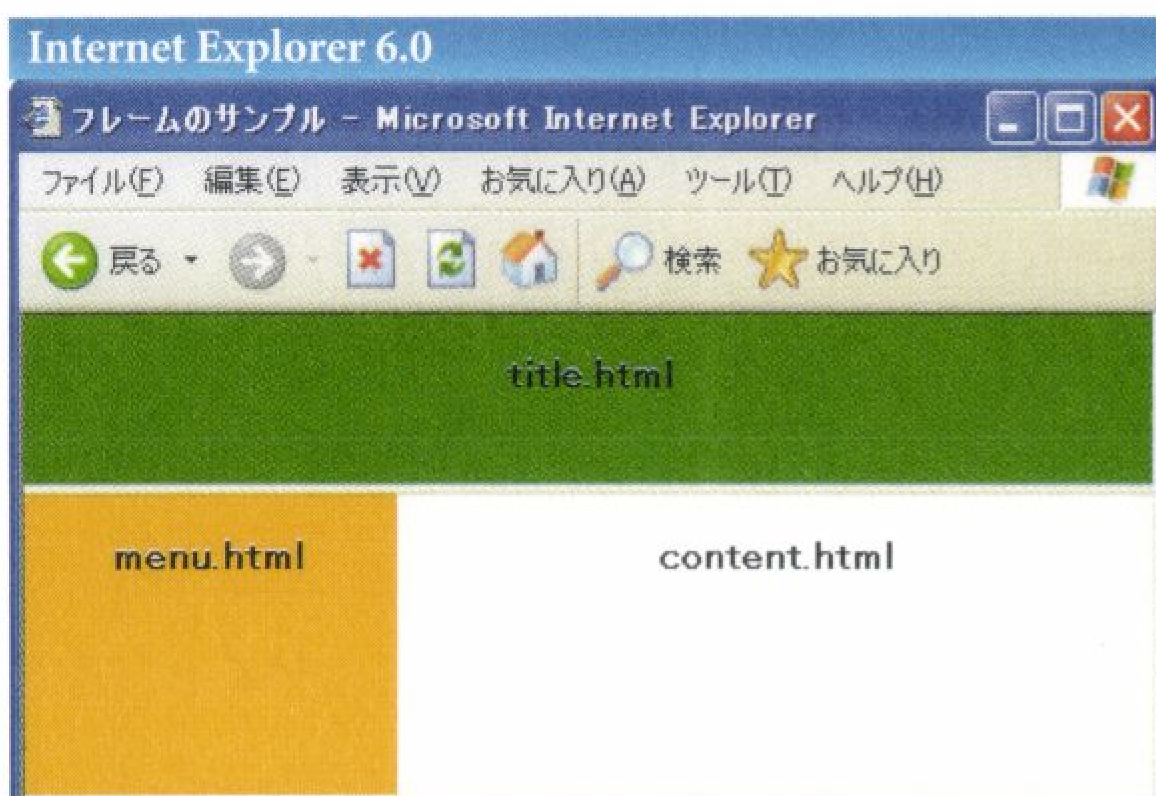
IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# フレームを区切る枠の表示・非表示を設定する

**<frame frameborder="枠の表示指定">**

| 【枠の表示指定】 | |
|---|---|
| 1 | 表示（デフォルト） |
| 0 | 非表示 |

Internet Explorer 6.0




Firefox 1.0




frameborder属性は、そのフレームと隣接するフレームとを区切る枠を表示するかどうかを設定します。
ここでいう枠とは、一般的に立体的に表示される枠のことで、これを非表示にしてもフレームとフレームの間の（平面的な）空間は残ったままになります。一方のフレームの枠が非表示に設定されていても、隣接する他方のフレームの枠が表示されるように設定されている場合には、その間の枠は表示されますので、注意してください。
一般的なブラウザでは、この属性をframeset要素にも指定できますが、HTML4.01の仕様ではframe要素に対してのみ指定できることになっています。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
 "http://www.w3.org/TR/html4/frameset.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>フレームのサンプル</title>
</head>
<frameset rows="70,*">
  <frame src="title.html">
  <frameset cols="150,*">
    <frame src="menu.html" frameborder="0">
    <frame src="content.html" frameborder="0">
  </frameset>
  <noframes>
  <body>
   ∫
  </body>
  </noframes>
</frameset>
</html>
```


<frameset frameborder="0">：「フレーム」の「フレームを区切る枠を完全に消す」(P.142)

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

# フレームを区切る枠を完全に消す

**<frameset frameborder="0" framespacing="0" border="0">～</frameset>**

Internet Explorer 6.0




Firefox 1.0




frame要素のframeborder属性を使用すると、フレームを区切る枠を非表示に設定することはできますが、フレームとフレームの間の枠が平面的になるだけで空間は残ってしまいます。それらも含めた枠を完全に消すためには、frameset要素にframeborder属性を指定した上で、さらにInternet Explorerの独自拡張であるframespacing属性と、Netscape Navigatorの独自拡張であるborder属性を指定する必要があります。
これらの独自拡張を利用すると、そのHTML文書は標準的な仕様に沿ったものではなくなり、しかも多くのブラウザでフレームのサイズが変更できなくなるなどの弊害も出てきますので、注意してください。

## Sample

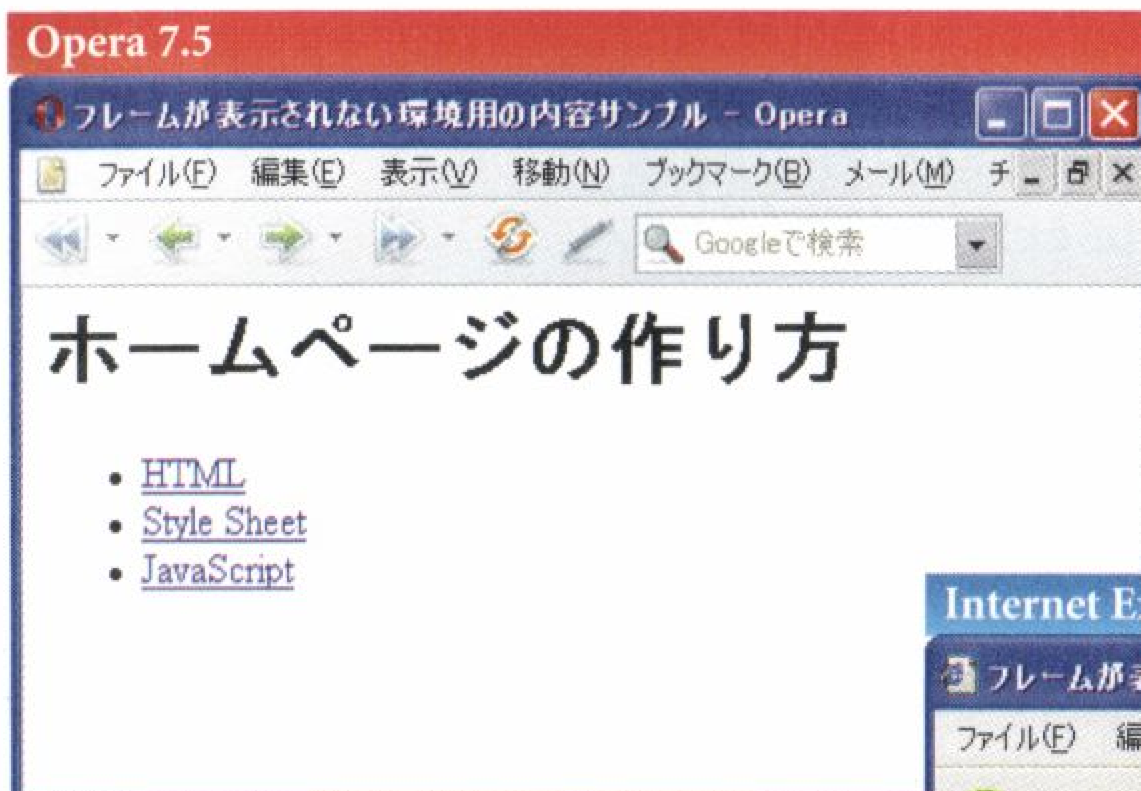
```
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=Shift_JIS">
<title>フレームのサンプル</title>
</head>
<frameset rows="70,*" frameborder="0" framespacing="0" border="0">
  <frame src="title.html">
  <frameset cols="150,*">
    <frame src="menu.html">
    <frame src="content.html">
  </frameset>
  <noframes>
  <body>
   ∫
  </body>
  </noframes>
</frameset>
</html>
```

# フレームが表示されない環境用の内容を入れる

## <noframes>～</noframes>



Opera 7.5




Internet Explorer 6.0




noframes要素は、フレームをサポートしていないブラウザを使っている場合やフレーム機能をオフにしている場合など、フレームが表現できない環境で表示させる内容を指定します。
この要素は、<frameset>～</frameset>の範囲の最初または最後にひとつだけ配置してください。<noframes>～</noframes>の中には、まずbody要素を配置して、その中に表示させたい内容を記述します。内容としては、「フレームに対応したブラウザでご覧ください」というようなものではなく、フレーム版の代わりとなる内容そのものや、各ページの説明とそのリンクなどを入れるようにしてください。

### Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" ↵
 "http://www.w3.org/TR/html4/frameset.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html; ↵
 charset=Shift_JIS">
<title>フレームが表示されない環境用の内容サンプル</title>
</head>
<frameset rows="70,*">
  <frame src="title.html" name="logo">
  <frameset cols="150,*">
```

```
    <frame src="menu.html" name="menu">
    <frame src="content.html" name="content">
  </frameset>
  <noframes>
    <body>
      <h1>ホームページの作り方</h1>
      <ul>
      <li><a href="ht.html">HTML</a></li>
      <li><a href="ss.html">Style Sheet</a></li>
      <li><a href="js.html">JavaScript</a></li>
      </ul>
    </body>
  </noframes>
</frameset>
</html>
```

## コラム

### テキストブラウザや音声ブラウザなどに対する配慮

テキストブラウザや音声ブラウザなどの中にも、フレームを取り扱うことができるものがあります。その場合、HTMLの内容から独自に情報を抜き出してフレームの構成をリンクとして示し、そこから各フレームを個別に参照できるようになっているものが多いようです。

ところが、リンクで各フレームを示す場合に、何を利用して示すかがブラウザごとに異なっており、frame要素のname属性またはtitle属性の値をリンクとして使用するものもあれば、frame要素で指定されている文書のtitle要素の内容をリンクとして使用するものもあります。

WAI(Web Accessibility Intaractive)のアクセシビリティ・ガイドラインでは、frame要素のtitle属性を使用してフレームのタイトルを付けることが推奨されています。しかし、現状では、各frame要素のname属性とtitle属性、各フレームの内容となる文書のtitle要素の3種類に対して、そのフレームの役割がわかるような適切なものを指定しておく必要があるようです。

# リンク先をどのフレームに表示するかを指定する

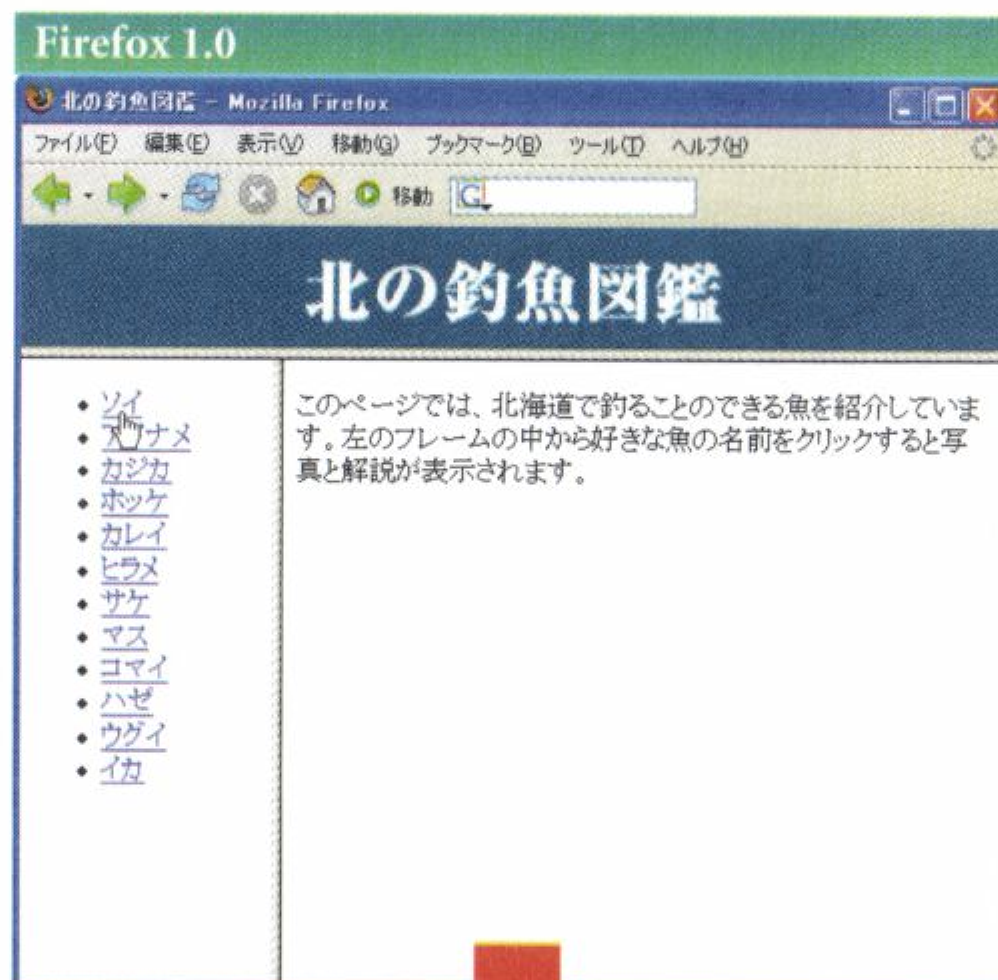
**<a href="*URL*" target="フレーム名">～</a>**

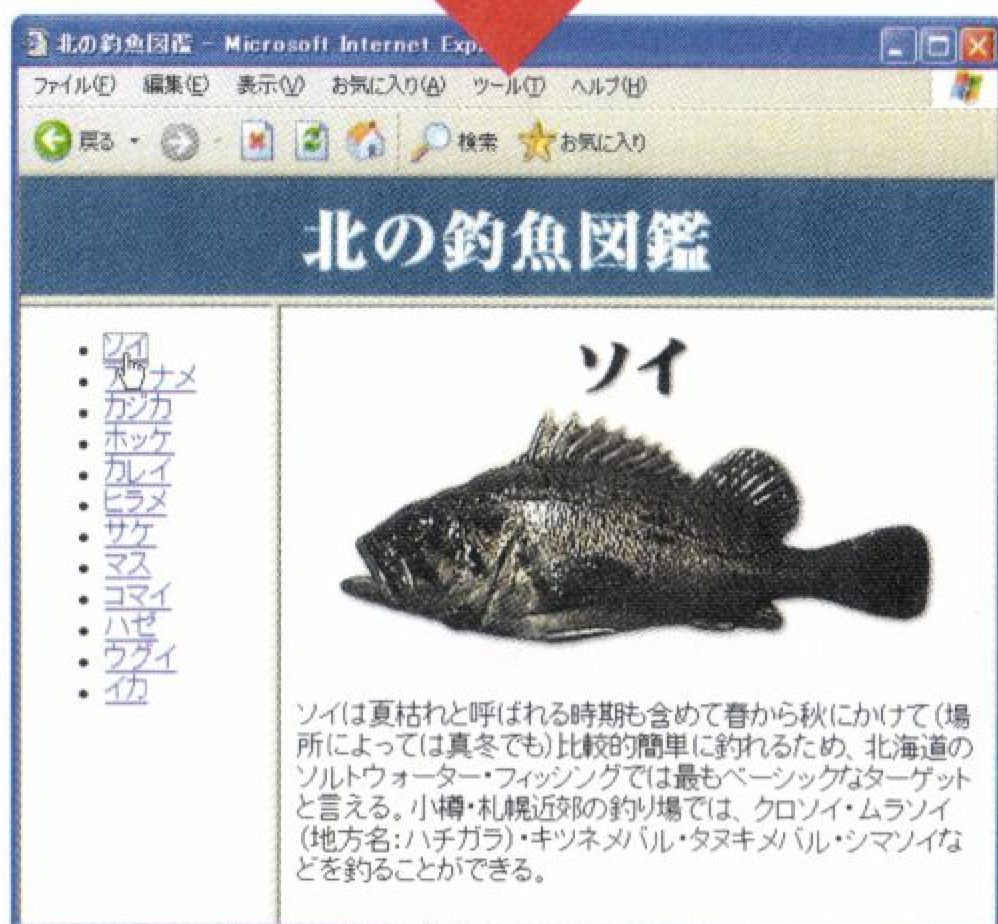


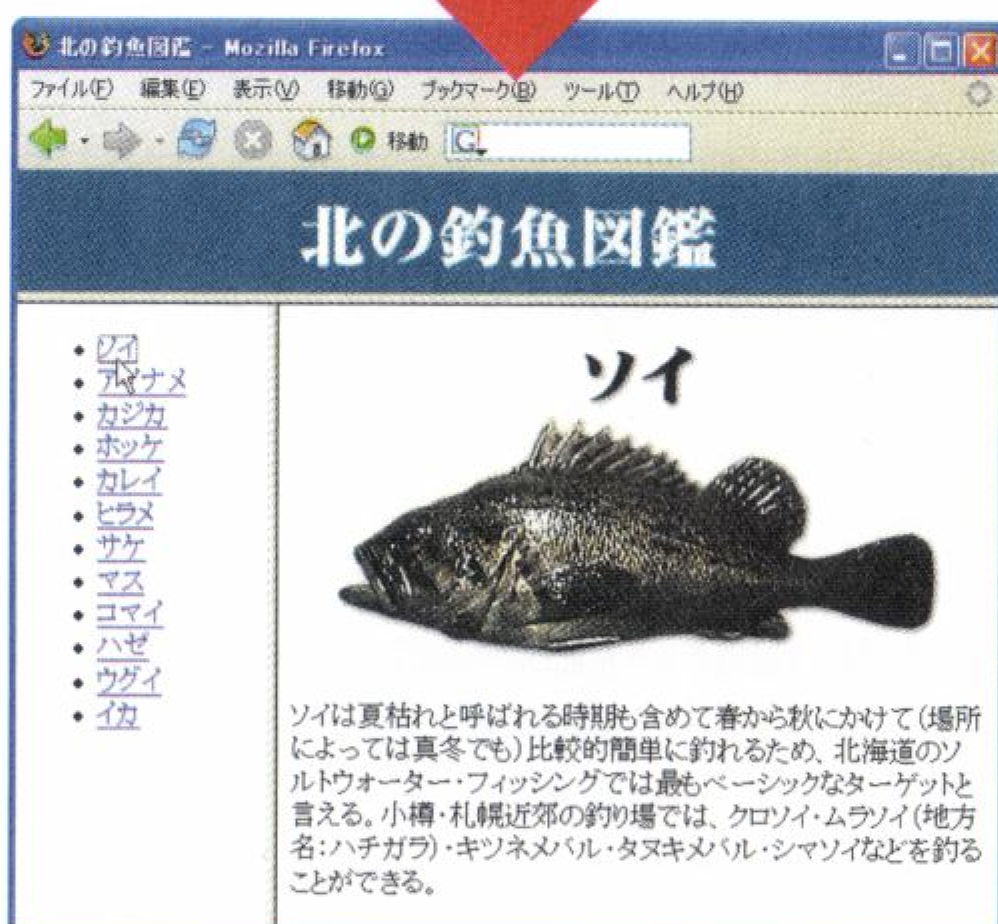
Internet Explorer 6.0




Firefox 1.0




フレーム内の文書で指定されているリンクは、そのままだとリンク先を同じフレーム内に表示します。これを他のフレームに表示させるようにしたい場合には、表示させたいフレーム(frame要素)のname属性で指定してあるフレームの名前を、target属性の値としてそのまま指定します。
また、フレーム名には「_top」のように「_」で始まる4つの特別な名前があります。これについては、TIPS「target属性の特別な4つの値」(P.62)を参照してください。

## Sample

**【フレームを定義している文書】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
 "http://www.w3.org/TR/html4/frameset.dtd">
<html lang="ja">
<head>
```

```
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>北の釣魚図鑑</title>
</head>
<frameset rows="70,*">
  <frame src="title.html" name="logo" scrolling="no">
  <frameset cols="150,*">
    <frame src="menu.html" name="menu">
    <frame src="content.html" name="content">
  </frameset>
  <noframes>
  <body>
   ∫
  </body>
  </noframes>
</frameset>
</html>
```

**【左側のフレームの文書（menu.html）】**

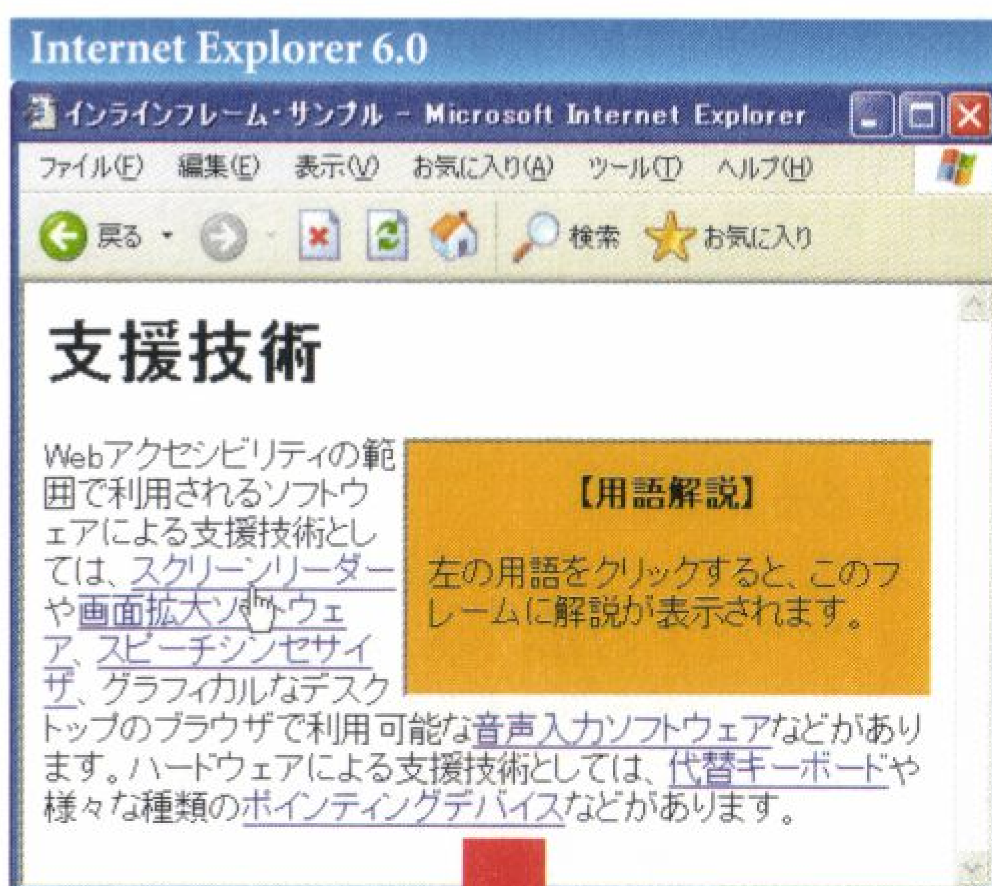
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>北の釣魚メニュー</title>
</head>
<body>
<ul>
<li><a href="soi.html" target="content">ソイ</a></li>
<li><a href="ainame.html" target="content">アイナメ</a></li>
<li><a href="kajika.html" target="content">カジカ</a></li>
<li><a href="hokke.html" target="content">ホッケ</a></li>
<li><a href="karei.html" target="content">カレイ</a></li>
<li><a href="hirame.html" target="content">ヒラメ</a></li>
<li><a href="sake.html" target="content">サケ</a></li>
<li><a href="masu.html" target="content">マス</a></li>
<li><a href="komai.html" target="content">コマイ</a></li>
<li><a href="haze.html" target="content">ハゼ</a></li>
<li><a href="ugui.html" target="content">ウグイ</a></li>
<li><a href="ika.html" target="content">イカ</a></li>
</ul>
</body>
</html>
```

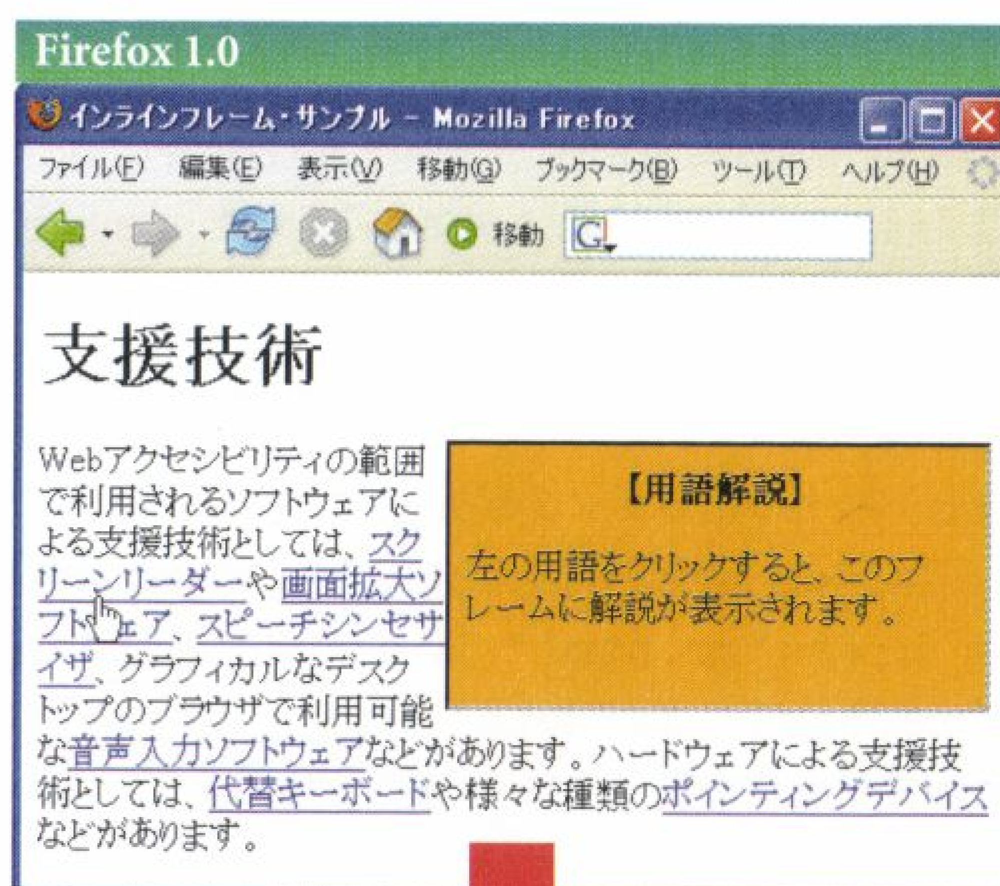
<base href="～">：「文書情報」の「基準URLを設定する」(P.24)
<a href="～" target="～">～</a>：「リンク」の「リンク先を別のウィンドウに表示する」(P.61)

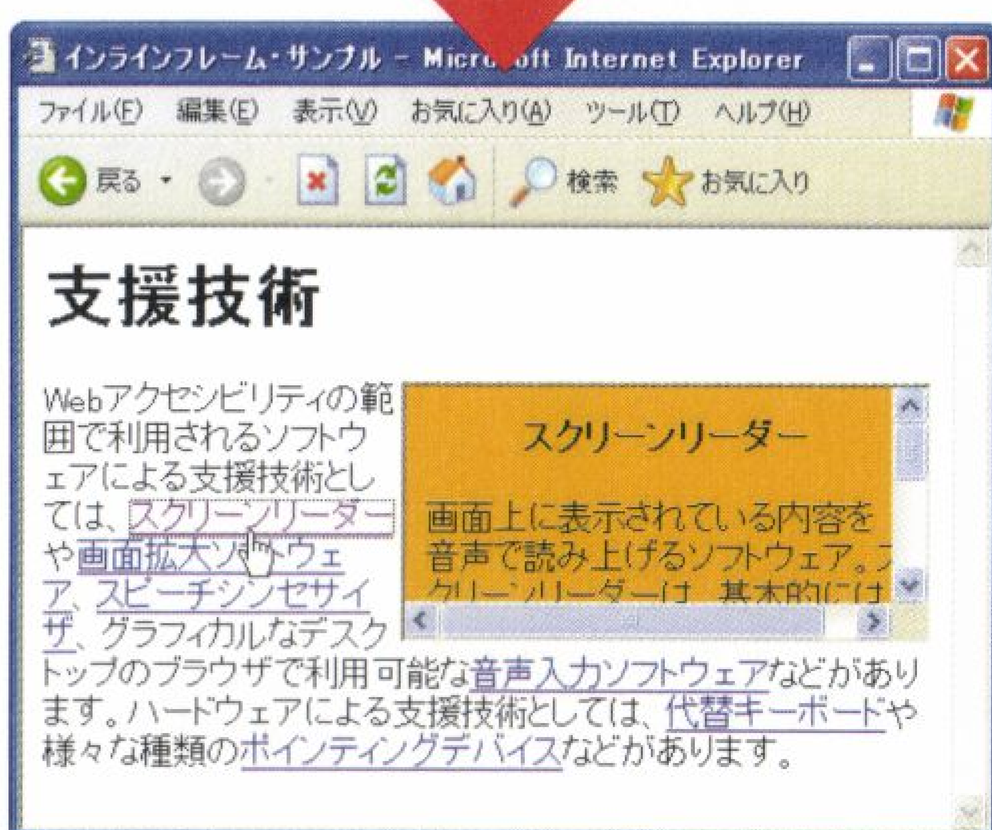
# インラインフレームを配置する

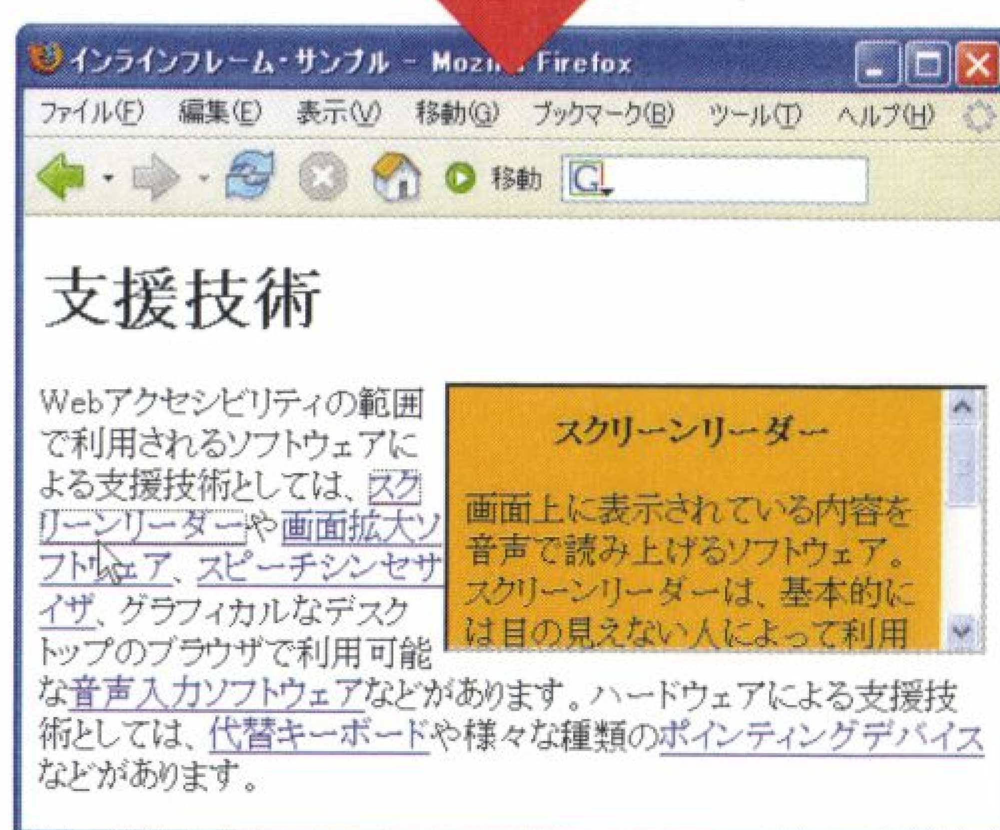
**<iframe src="内容のURL" name="フレーム名">～</iframe>**

| 【その他に指定可能な属性】 | |
|---|---|
| width | フレームの幅（ピクセル・%） |
| height | フレームの高さ（ピクセル・%） |
| marginwidth | フレーム内の左右のマージン（ピクセル） |
| marginheight | フレーム内の上下のマージン（ピクセル） |
| scrolling | スクロール（auto：自動／yes：あり／no：なし） |
| frameborder | フレーム枠の表示・非表示（1：表示／0：非表示） |
| align | 配置位置（left・right・top・middle・bottom） |




iframe要素は、ウィンドウを分割する形式のフレームではなく、ウィンドウの中に独立して表示されるインラインフレームを配置します。フレーム内には、src属性で指定された内容が表示されます。

<iframe>～</iframe>の間には、このフレームをサポートしていないブラウザや、フレームを表示しないように設定している場合に表示させたい内容を指定しておきます。

このフレームはframeset要素で定義されるフレームとは異なり、<!DOCTYPE>が「Transitional」の場合でも使用することができます。align属性でleftまたはrightを指定すると、インラインフレームが左または右に配置されてテキストが回り込むようになります。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" 
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html; 
 charset=Shift_JIS">
<title>インラインフレーム・サンプル</title>
</head>
<body>
<h1>支援技術</h1>
<p>
<iframe src="kaisetsu.html" name="term" width="250" height="120" 
 align="right">
    …インラインフレームが表示されない環境用の内容…
</iframe>
Webアクセシビリティの範囲で利用されるソフトウェアによる支援技術と
しては、<a href="sr.html" target="term">スクリーンリーダー</a>や
<a href="sm.html" target="term">画面拡大ソフトウェア</a>、<a href
="ss.html" target="term">スピーチシンセサイザ</a>、グラフィカルな
デスクトップのブラウザで利用可能な<a href="vi.html" target="term">
音声入力ソフトウェア</a>などがあります。ハードウェアによる支援技術
としては、<a href="ak.html" target="term">代替キーボード</a>や様々
な種類の<a href="pd.html" target="term">ポインティングデバイス</a>
などがあります。
<br clear="right">
</p>
</body>
</html>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X Opera7 Opera6 Safari IE5-mac IE4-mac

# HTMLにスクリプトを組み込む

**<script type="*MIME*タイプ">～</script>**
**<script type="*MIME*タイプ" language="言語名" src="*URL*">～</script>**

| | |
|---|---|
| MIMEタイプ | スクリプト言語のMIMEタイプ（例：text/javascript） |
| 言語名 | スクリプト言語の名前（例：JavaScript1.2） |
| URL | スクリプトを記述した別ファイルのURL |

script要素は、HTML文書内にスクリプトを記述する場合に使用し、スクリプト言語はこの要素の範囲内に記述します。
この時、この要素に未対応のブラウザがスクリプト部分を画面に表示してしまうことを避けるために、通常はスクリプト全体をHTMLでのコメントにしておきます（コメントの終了部分が、HTMLの場合とは多少異なることに注意してください）。この要素は、<head>～</head>と<body>～</body>の範囲内の任意の位置に置くことができます。
type属性はHTML4.0から採用された属性で、仕様上は必ず指定することになっています。language属性は古くから利用されている属性で、バージョンに依存したスクリプトを使用したい場合などに利用されていますが、HTML4.0では非推奨属性とされています。src属性は、スクリプトを別ファイルに記述しておいて、それを呼び出して使いたい場合に使用します。

## Sample

```
<script type="text/javascript">
<!--
if (self != top)
  top.location.href = self.location.href;
// -->
</script>
```


<meta http-equiv="～" content="～">：「文書情報」の「スタイルシートやスクリプトの言語を示す」(P.21)
JavaScript：「JavaScriptについて」の「JavaScriptの記述法（<script>の使い方）」(P.299)
JavaScript：「JavaScriptについて」の「JavaScriptの記述法（JavaScriptの外部呼び込み方法）」(P.302)

# スクリプトが実行されない環境用の内容を入れる

## <noscript>～</noscript>

noscript要素は、スクリプトが実行できない場合に、代わりに表示させる内容を指定します。
したがって、スクリプトが実行可能な場合には、ここに記入した内容は表示されません。この要素は、<body>～</body>の範囲内に配置するようにしてください。また、この要素の内容としては、「スクリプト対応のブラウザで見てください」というものではなく、スクリプトが利用できる場合と同じような意味を持つ内容(または、そのようなページへのリンク)を入れるようにしてください。

### Sample

```
<noscript>
<p>
スクリプトが利用できない環境の方は、
<a href="acs.html">アクセシブル版</a>
をご利用ください。
</p>
</noscript>
```

 JavaScript：「JavaScriptについて」の「JavaScriptの記述法(<noscript>の使い方)」(P.302)

# HTML4.01 全要素・属性一覧

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| a | | リンク（アンカー） | Strict | | | |
| | href | リンク先URL | Strict | | | 55,※1 |
| | charset | リンク先の文字コード | Strict | | | |
| | hreflang | リンク先の言語コード | Strict | | | |
| | type | リンク先のMIMEタイプ | Strict | | | |
| | name | 名前 | Strict | | | 56,58 |
| | rel | この文書から見た関係 | Strict | | | |
| | rev | リンク先から見た関係 | Strict | | | |
| | shape | イメージマップの領域の形状 | Strict | | | |
| | coords | イメージマップの領域の座標 | Strict | | | |
| | tabindex | Tab移動順 | Strict | | | |
| | accesskey | ショートカットキー | Strict | | | |
| | target | 表示先のフレーム・ウィンドウ名 | Transitional | | | 61,145 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| abbr | | 略語 | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 省略していない元の言葉 | Strict | | | 35 |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| acronym | | 頭字語 | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 省略していない元の言葉 | Strict | | | 35 |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| address | | 連絡先 | Strict | | | 30 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |

※1：56,58,63,145

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| applet | | JAVAアプレット | Transitional | 非推奨 | | |
| | width | 幅 | Transitional | | 必須 | 112 |
| | height | 高さ | Transitional | | 必須 | 112 |
| | code | クラスファイル名 | Transitional | | | 112 |
| | object | シリアライズされたデータ名 | Transitional | | | |
| | codebase | 基準URL | Transitional | | | |
| | archive | JAVAアーカイブファイルのURL | Transitional | | | |
| | alt | 代替テキスト | Transitional | | | |
| | name | 名前 | Transitional | | | |
| | align | テキストとの位置関係 | Transitional | 非推奨 | | |
| | hspace | 左右の空間 | Transitional | 非推奨 | | |
| | vspace | 上下の空間 | Transitional | 非推奨 | | |
| | class | クラス名 | Transitional | | | |
| | id | ID名 | Transitional | | | |
| | title | 補足情報 | Transitional | | | |
| | style | スタイルシート | Transitional | | | |
| area | | イメージマップの領域指定 | Strict | | | |
| | alt | 代替テキスト | Strict | | 必須 | 103 |
| | shape | 領域の形状 | Strict | | | 103 |
| | coords | 領域の座標 | Strict | | | 103,104 |
| | href | リンク先のURL | Strict | | | 103 |
| | nohref | リンクが無いことを示す | Strict | | | |
| | tabindex | Tab移動順 | Strict | | | |
| | accesskey | ショートカットキー | Strict | | | |
| | target | 表示先のフレーム・ウィンドウ名 | Transitional | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| b | | 太字 | Strict | | | 44 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| base | | 基準URL | Strict | | | |
| | href | 基準URL | Strict | | | 24 |
| | target | 表示先のフレーム・ウィンドウ名 | Transitional | | | 24 |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| basefont | | 基準フォント・色・サイズ | Transitional | 非推奨 | | |
| | face | フォント名 | Transitional | 非推奨 | | |
| | color | 色 | Transitional | 非推奨 | | |
| | size | フォントサイズ | Transitional | 非推奨 | 必須 | 54 |
| | id | ID名 | Transitional | | | |
| bdo | | 文字表記の方向 | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | 必須 | |
| big | | 大きな文字 | Strict | | | 52 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| blockquote | | ブロックレベルの引用文 | Strict | | | 33 |
| | cite | 出典URL | Strict | | | 33 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| body | | 文書本体 | Strict | | | 11 |
| | text | 文字色 | Transitional | 非推奨 | | 12 |
| | link | 未キャッシュのリンク部分の色 | Transitional | 非推奨 | | 59 |
| | vlink | キャッシュ済みのリンク部分の色 | Transitional | 非推奨 | | 59 |
| | alink | リンク部分を選択した時の色 | Transitional | 非推奨 | | 59 |
| | bgcolor | 背景色 | Transitional | 非推奨 | | 13 |
| | background | 背景画像のURL | Transitional | 非推奨 | | 14 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| br | | 改行 | Strict | | | 42 |
| | clear | 回り込みの解除 | Transitional | 非推奨 | | 96,106 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| button | | ボタン | Strict | | | |
| | name | 名前 | Strict | | | 119 |
| | type | 形式 | Strict | | | 119 |
| | value | 値 | Strict | | | 119 |
| | disabled | 選択・変更不可 | Strict | | | |
| | tabindex | Tab移動順 | Strict | | | |
| | accesskey | ショートカットキー | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| caption | | 表のタイトル | Strict | | | 72 |
| | align | 表示位置 | Transitional | 非推奨 | | 72 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| center | | 中央揃え | Transitional | 非推奨 | | 47 |
| | class | クラス名 | Transitional | | | |
| | id | ID名 | Transitional | | | |
| | title | 補足情報 | Transitional | | | |
| | style | スタイルシート | Transitional | | | |
| | lang | 言語コード | Transitional | | | |
| | dir | 文字表記の方向 | Transitional | | | |
| cite | | 出典・参照先 | Strict | | | 34 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| code | | ソースコード | Strict | | | 39 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| col | | 表の縦列の属性値共有 | Strict | | | |
| | span | 対象とする縦列数 | Strict | | | 92 |
| | width | 縦列の幅 | Strict | | | 92 |
| | align | 行揃え | Strict | | | 79 |
| | valign | 縦方向の位置揃え | Strict | | | 79 |
| | char | 位置を揃える文字 | Strict | | | |
| | charoff | 位置を揃える文字までの距離 | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| colgroup | | 表の縦列のグループ化 | Strict | | | |
| | span | グループ化する縦列数 | Strict | | | 90 |
| | width | 縦列の幅 | Strict | | | 90 |
| | align | 行揃え | Strict | | | 79 |
| | valign | 縦方向の位置揃え | Strict | | | 79 |
| | char | 位置を揃える文字 | Strict | | | |
| | charoff | 位置を揃える文字までの距離 | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| dd | | dl要素の説明部分 | Strict | | | 69 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| del | | 削除部分 | Strict | | | |
| | cite | 削除理由のURL | Strict | | | 37 |
| | datetime | 更新日時 | Strict | | | 37 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| dfn | | 定義対象の用語 | Strict | | | 38 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| dir | | ディレクトリリスト | Transitional | 非推奨 | | |
| | compact | 詰めて表示 | Transitional | 非推奨 | | |
| | class | クラス名 | Transitional | | | |
| | id | ID名 | Transitional | | | |
| | title | 補足情報 | Transitional | | | |
| | style | スタイルシート | Transitional | | | |
| | lang | 言語コード | Transitional | | | |
| | dir | 文字表記の方向 | Transitional | | | |
| div | | ブロックレベルの範囲の設定 | Strict | | | 16 |
| | align | 行揃え | Transitional | 非推奨 | | 48 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| dl | | 用語と説明のリスト | Strict | | | 69 |
| | compact | 詰めて表示 | Transitional | 非推奨 | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| dt | | dl要素の用語部分 | Strict | | | 69 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| em | | 強調 | Strict | | | 31 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| fieldset | | フォームの内容のグループ化 | Strict | | | 131 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| font | | フォントの種類・色・サイズ | Transitional | 非推奨 | | |
| | face | フォント名 | Transitional | 非推奨 | | 50 |
| | color | 色 | Transitional | 非推奨 | | 49 |
| | size | フォントサイズ | Transitional | 非推奨 | | 51,52 |
| | class | クラス名 | Transitional | | | |
| | id | ID名 | Transitional | | | |
| | title | 補足情報 | Transitional | | | |
| | style | スタイルシート | Transitional | | | |
| | lang | 言語コード | Transitional | | | |
| | dir | 文字表記の方向 | Transitional | | | |
| form | | 入力フォーム | Strict | | | |
| | action | データの処理をするURL | Strict | | 必須 | 113,133 |
| | method | HTTPメソッド(get／post) | Strict | | | 113,133 |
| | enctype | データを送信する際のMIMEタイプ | Strict | | | 113,133 |
| | accept-charset | 受け付け可能な文字コード | Strict | | | |
| | accept | 受け付け可能なMIMEタイプ | Strict | | | |
| | name | 名前 | Strict | | | |
| | target | 表示先のフレーム・ウィンドウ名 | Transitional | | | 113 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| frame | | フレームの内容指定 | Frameset | | | |
| | src | フレームの内容のURL | Frameset | | | 135 |
| | name | フレームの名前 | Frameset | | | 135 |
| | longdesc | フレームの詳しい説明のURL | Frameset | | | |
| | marginwidth | フレーム内の左右のマージン | Frameset | | | 138 |
| | marginheight | フレーム内の上下のマージン | Frameset | | | 138 |
| | noresize | サイズの変更不可 | Frameset | | | 138 |
| | scrolling | スクロールの制御 | Frameset | | | 138 |
| | frameborder | 枠の表示 | Frameset | | | 140 |
| | class | クラス名 | Frameset | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| | id | ID名 | Frameset | | | |
| | title | 補足情報 | Frameset | | | |
| | style | スタイルシート | Frameset | | | |
| frameset | | フレームの定義 | Frameset | | | |
| | rows | フレームの各高さ | Frameset | | | 135 |
| | cols | フレームの各幅 | Frameset | | | 135 |
| | class | クラス名 | Frameset | | | |
| | id | ID名 | Frameset | | | |
| | title | 補足情報 | Frameset | | | |
| | style | スタイルシート | Frameset | | | |
| h1～h6 | | 見出し | Strict | | | 27 |
| | align | 行揃え | Transitional | 非推奨 | | 48 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| head | | 文書情報 | Strict | | | 11 |
| | profile | メタデータ・プロファイルのURL | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| hr | | 横罫線 | Strict | | | 43 |
| | align | 横方向の配置位置 | Transitional | 非推奨 | | 43 |
| | size | 太さ | Transitional | 非推奨 | | 43 |
| | width | 長さ | Transitional | 非推奨 | | 43 |
| | noshade | 平面的に表示 | Transitional | 非推奨 | | 43 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| html | | 最上位要素 | Strict | | | 11 |
| | version | HTMLのバージョン | Transitional | 非推奨 | | |
| | lang | 言語コード | Strict | | | 26 |
| | dir | 文字表記の方向 | Strict | | | |
| i | | イタリック | Strict | | | 44 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| iframe | | インラインフレーム | Transitional | | | |
| | src | フレームの内容のURL | Transitional | | | 147 |
| | name | フレームの名前 | Transitional | | | 147 |
| | longdesc | フレームの詳しい説明のURL | Transitional | | | |
| | width | フレームの幅 | Transitional | | | |
| | height | フレームの高さ | Transitional | | | |
| | marginwidth | フレーム内の左右のマージン | Transitional | | | |
| | marginheight | フレーム内の上下のマージン | Transitional | | | |
| | scrolling | スクロールの制御 | Transitional | | | |
| | frameborder | 枠の表示 | Transitional | | | |
| | align | フレームとテキストの位置関係 | Transitional | 非推奨 | | |
| | class | クラス名 | Transitional | | | |
| | id | ID名 | Transitional | | | |
| | title | 補足情報 | Transitional | | | |
| | style | スタイルシート | Transitional | | | |
| img | | 画像 | Strict | | | |
| | src | 画像のURL | Strict | | 必須 | 99 |
| | alt | 代替テキスト | Strict | | 必須 | 99,108 |
| | longdesc | 詳しい情報のURL | Strict | | | |
| | width | 幅 | Strict | | | 99 |
| | height | 高さ | Strict | | | 99 |
| | usemap | イメージマップの指定 | Strict | | | 103 |
| | ismap | サーバーサイド・イメージマップ | Strict | | | |
| | name | 名前 | Strict | | | |
| | align | テキストとの位置関係 | Transitional | 非推奨 | | 102,105 |
| | border | 枠線の太さ | Transitional | 非推奨 | | 100 |
| | hspace | 左右の空間 | Transitional | 非推奨 | | 107 |
| | vspace | 上下の空間 | Transitional | 非推奨 | | 107 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| input | | 入力・選択用部品とボタン | Strict | | | |
| | type | 表示形式 | Strict | | | ※2 |
| | name | 名前 | Strict | | | ※3 |
| | value | 初期値／ラベル文字 | Strict | | | ※4 |
| | size | 表示幅 | Strict | | | 120,122 |
| | maxlength | 最大文字数 | Strict | | | 120,122 |
| | checked | 選択状態 | Strict | | | 124 |
| | disabled | 選択・変更不可 | Strict | | | |
| | readonly | 変更不可 | Strict | | | |

※2：115～117,120,122～125,130／※3：115,117,120,122～125,130／※4：115～117,120,122～125

リファレンス

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| | accept | 受け付け可能なMIMEタイプ | Strict | | | 130 |
| | tabindex | Tab移動順 | Strict | | | |
| | accesskey | ショートカットキー | Strict | | | |
| | src | 画像のURL | Strict | | | |
| | alt | 画像の代替テキスト | Strict | | | |
| | usemap | イメージマップの指定 | Strict | | | |
| | ismap | サーバーサイド・イメージマップ | Strict | | | |
| | align | テキストとの位置関係 | Transitional | 非推奨 | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| ins | | 追加部分 | Strict | | | |
| | cite | 追加理由のURL | Strict | | | 36 |
| | datetime | 更新日時 | Strict | | | 36 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| isindex | | 検索用入力フィールド | Transitional | 非推奨 | | |
| | prompt | 入力フィールドのラベル | Transitional | 非推奨 | | |
| | class | クラス名 | Transitional | | | |
| | id | ID名 | Transitional | | | |
| | title | 補足情報 | Transitional | | | |
| | style | スタイルシート | Transitional | | | |
| | lang | 言語コード | Transitional | | | |
| | dir | 文字表記の方向 | Transitional | | | |
| kbd | | 入力テキスト | Strict | | | 39 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| label | | フォームの部品のラベル | Strict | | | 134 |
| | for | 関連フォーム部品のID | Strict | | | 132 |
| | accesskey | ショートカットキー | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| legend | | fieldset要素のラベル | Strict | | | |
| | accesskey | ショートカットキー | Strict | | | |
| | align | ラベルの表示位置 | Transitional | 非推奨 | | 131 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| li | | リスト項目 | Strict | | | 64,66 |
| | type | マークの種類 | Transitional | 非推奨 | | 65,67 |
| | value | 番号 | Transitional | 非推奨 | | 68 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| link | | 関連ファイル | Strict | | | |
| | rel | この文書から見た関係 | Strict | | | 22 |
| | rev | 関連文書から見た関係 | Strict | | | 22 |
| | href | 関連文書のURL | Strict | | | 22 |
| | hreflang | 関連文書の言語コード | Strict | | | |
| | charset | 関連文書の文字コード | Strict | | | |
| | type | 関連文書のMIMEタイプ | Strict | | | |
| | media | 関連文書のメディアタイプ | Strict | | | |
| | target | 表示先のフレーム・ウィンドウ名 | Transitional | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| map | | イメージマップの定義 | Strict | | | |
| | name | 名前 | Strict | | 必須 | 103 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| menu | | メニューリスト | Transitional | 非推奨 | | |
| | compact | 詰めて表示 | Transitional | 非推奨 | | |
| | class | クラス名 | Transitional | | | |
| | id | ID名 | Transitional | | | |
| | title | 補足情報 | Transitional | | | |
| | style | スタイルシート | Transitional | | | |
| | lang | 言語コード | Transitional | | | |
| | dir | 文字表記の方向 | Transitional | | | |
| meta | | メタ情報 | Strict | | | |
| | content | メタデータのプロパティの値 | Strict | | 必須 | 19,※5 |
| | name | メタデータのプロパティ名 | Strict | | | 20 |
| | http-equiv | HTTPヘッダ用プロパティ名 | Strict | | | 19,※6 |
| | scheme | プロパティの値の形式 | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| noframes | | フレームが使えない環境用 | Frameset | | | 143 |
| | class | クラス名 | Frameset | | | |
| | id | ID名 | Frameset | | | |
| | title | 補足情報 | Frameset | | | |
| | style | スタイルシート | Frameset | | | |
| | lang | 言語コード | Frameset | | | |
| | dir | 文字表記の方向 | Frameset | | | |
| noscript | | スクリプトが使えない環境用 | Strict | | | 150 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| object | | オブジェクト | Strict | | | |
| | data | データのURL | Strict | | | 109 |
| | type | データのMIMEタイプ | Strict | | | 109 |
| | classid | 実行プログラムのURL | Strict | | | |
| | codetype | 実行プログラムのMIMEタイプ | Strict | | | |
| | archive | アーカイブのURL | Strict | | | |
| | codebase | 基準URL | Strict | | | |
| | width | 幅 | Strict | | | 109 |
| | height | 高さ | Strict | | | 109 |
| | usemap | イメージマップの指定 | Strict | | | |
| | declare | 宣言のみで実行はしない | Strict | | | |
| | standby | ロード中のメッセージ | Strict | | | |
| | name | 名前 | Strict | | | |
| | tabindex | Tab移動順 | Strict | | | |

※5：20,21,25／※6：21,25

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| | align | テキストとの位置関係 | Transitional | 非推奨 | | |
| | border | 枠線の太さ | Transitional | 非推奨 | | |
| | hspace | 左右の空間 | Transitional | 非推奨 | | |
| | vspace | 上下の空間 | Transitional | 非推奨 | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| ol | | 番号付きのリスト | Strict | | | 66 |
| | type | 番号の種類 | Transitional | 非推奨 | | 67 |
| | compact | 詰めて表示 | Transitional | 非推奨 | | |
| | start | 開始番号 | Transitional | 非推奨 | | 68 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| optgroup | | メニュー選択肢のグループ化 | Strict | | | |
| | label | 第一階層用の選択肢 | Strict | | 必須 | 127 |
| | disabled | 選択・変更不可 | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| option | | メニューの選択肢 | Strict | | | |
| | selected | 選択状態 | Strict | | | 126,129 |
| | value | 値 | Strict | | | 126,129 |
| | label | 階層メニュー用の選択肢 | Strict | | | 127 |
| | disabled | 選択・変更不可 | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| p | | 段落（パラグラフ） | Strict | | | 29 |
| | align | 行揃え | Transitional | 非推奨 | | 48 |
| | class | クラス名 | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| param | | パラメータ | Strict | | | |
| | name | 名前 | Strict | | 必須 | 112 |
| | value | パラメータの値 | Strict | | | 112 |
| | valuetype | パラメータの値の形式 | Strict | | | |
| | type | パラメータの値のMIMEタイプ | Strict | | | |
| | id | ID名 | Strict | | | |
| pre | | 整形済みテキスト | Strict | | | 46 |
| | width | 表示文字数 | Transitional | 非推奨 | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| q | | インラインの引用文 | Strict | | | 31,32 |
| | cite | 出典URL | Strict | | | 32 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| s | | 取消線 | Transitional | 非推奨 | | 44 |
| | class | クラス名 | Transitional | | | |
| | id | ID名 | Transitional | | | |
| | title | 補足情報 | Transitional | | | |
| | style | スタイルシート | Transitional | | | |
| | lang | 言語コード | Transitional | | | |
| | dir | 文字表記の方向 | Transitional | | | |
| samp | | 出力サンプル | Strict | | | 39 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| script | | スクリプト | Strict | | | |
| | type | スクリプト言語のMIMEタイプ | Strict | | 必須 | 149 |
| | src | スクリプトファイルのURL | Strict | | | 149 |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| | charset | スクリプトファイルの文字コード | Strict | | | |
| | defer | 表示される内容を生成しない | Strict | | | |
| | language | スクリプト言語名 | Transitional | 非推奨 | | 149 |
| select | | メニュー | Strict | | | |
| | name | 名前 | Strict | | | 126,129 |
| | size | リストボックスの表示行数 | Strict | | | 129 |
| | multiple | 複数選択可 | Strict | | | 129 |
| | disabled | 選択・変更不可 | Strict | | | |
| | tabindex | Tab移動順 | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| small | | 小さな文字 | Strict | | | 52 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| span | | インラインの範囲の設定 | Strict | | | 16 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| strike | | 取消線 | Transitional | 非推奨 | | 44 |
| | class | クラス名 | Transitional | | | |
| | id | ID名 | Transitional | | | |
| | title | 補足情報 | Transitional | | | |
| | style | スタイルシート | Transitional | | | |
| | lang | 言語コード | Transitional | | | |
| | dir | 文字表記の方向 | Transitional | | | |
| strong | | より強い強調 | Strict | | | 31 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| style | | スタイルシート | Strict | | | |
| | type | スタイルシート言語のMIMEタイプ | Strict | | 必須 | |
| | media | 出力対象のメディアタイプ | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| sub | | 下付文字 | Strict | | | 44 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| sup | | 上付文字 | Strict | | | 44 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| table | | 表 | Strict | | | 70 |
| | summary | 表の概要 | Strict | | | |
| | width | 幅 | Strict | | | 73 |
| | border | 外枠の太さ | Strict | | | 85 |
| | frame | 外枠の表示形式 | Strict | | | 86 |
| | rules | セルを区切る線の表示形式 | Strict | | | 87 |
| | cellspacing | セルの間隔 | Strict | | | 77 |
| | cellpadding | セルの枠と内容の間隔 | Strict | | | 78 |
| | align | 表示位置 | Transitional | 非推奨 | | 94,95 |
| | bgcolor | 背景色 | Transitional | 非推奨 | | 83 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| tbody | | 表本体 | Strict | | | 88 |
| | align | 行揃え | Strict | | | 79 |
| | valign | 縦方向の位置揃え | Strict | | | 79 |
| | char | 位置を揃える文字 | Strict | | | |
| | choff | 位置を揃える文字間での距離 | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| td | | データセル | Strict | | | 70 |
| | rowspan | 下方向に連結するセルの数 | Strict | | | 75 |
| | colspan | 右方向に連結するセルの数 | Strict | | | 75 |
| | align | 行揃え | Strict | | | 79 |
| | valign | 縦方向の位置揃え | Strict | | | 79 |
| | char | 位置を揃える文字 | Strict | | | |
| | charoff | 位置を揃える文字までの距離 | Strict | | | |
| | abbr | 見出しセルの省略形 | Strict | | | |
| | axis | 見出しセルの分類名 | Strict | | | |
| | headers | 関連する見出しセルのID名 | Strict | | | |
| | scope | 見出しセルの対象となる範囲 | Strict | | | |
| | nowrap | 改行の禁止 | Transitional | 非推奨 | | 81 |
| | bgcolor | 背景色 | Transitional | 非推奨 | | 83 |
| | width | 幅 | Transitional | 非推奨 | | 74 |
| | height | 高さ | Transitional | 非推奨 | | 74 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| textarea | | 複数行の入力フィールド | Strict | | | |
| | rows | 行数 | Strict | | 必須 | 121 |
| | cols | 幅（文字数） | Strict | | 必須 | 121 |
| | name | 名前 | Strict | | | 121 |
| | disabled | 選択・変更不可 | Strict | | | |
| | readonly | 変更不可 | Strict | | | |
| | tabindex | Tab移動順 | Strict | | | |
| | accesskey | ショートカットキー | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| tfoot | | 表フッタ | Strict | | | 88 |
| | align | 行揃え | Strict | | | 79 |
| | valign | 縦方向の位置揃え | Strict | | | 79 |
| | char | 位置を揃える文字 | Strict | | | |
| | chroff | 位置を揃える文字間での距離 | Strict | | | |
| | class | クラス名 | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| th | | 見出しセル | Strict | | | 70 |
| | rowspan | 下方向に連結するセルの数 | Strict | | | 75 |
| | colspan | 右方向に連結するセルの数 | Strict | | | 75 |
| | align | 行揃え | Strict | | | 79 |
| | valign | 縦方向の位置揃え | Strict | | | 79 |
| | char | 位置を揃える文字 | Strict | | | |
| | charoff | 位置を揃える文字までの距離 | Strict | | | |
| | abbr | 見出しセルの省略形 | Strict | | | |
| | axis | 見出しセルの分類名 | Strict | | | |
| | headers | 関連する見出しセルのID名 | Strict | | | |
| | scope | 見出しセルの対象となる範囲 | Strict | | | |
| | nowrap | 改行の禁止 | Transitional | 非推奨 | | 81 |
| | bgcolor | 背景色 | Transitional | 非推奨 | | 83 |
| | width | 幅 | Transitional | 非推奨 | | 74 |
| | height | 高さ | Transitional | 非推奨 | | 74 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| thead | | 表ヘッダ | Strict | | | 88 |
| | align | 行揃え | Strict | | | 79 |
| | valign | 縦方向の位置揃え | Strict | | | 79 |
| | char | 位置を揃える文字 | Strict | | | |
| | chroff | 位置を揃える文字間での距離 | Strict | | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| title | | 文書タイトル | Strict | | | 11,18 |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| tr | | 表の横一列 | Strict | | | 70 |
| | align | 行揃え | Strict | | | 79 |
| | valign | 縦方向の位置揃え | Strict | | | 79 |
| | char | 位置を揃える文字 | Strict | | | |

| 要素名 | 属性名 | 説明 | DTDの種類 | 非推奨 | 必須属性 | 参照 |
|---|---|---|---|---|---|---|
| | charoff | 位置を揃える文字までの距離 | Strict | | | |
| | bgcolor | 背景色 | Transitional | 非推奨 | | 83 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| tt | | 等幅フォント | Strict | | | 44 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| u | | 下線 | Transitional | 非推奨 | | 44 |
| | class | クラス名 | Transitional | | | |
| | id | ID名 | Transitional | | | |
| | title | 補足情報 | Transitional | | | |
| | style | スタイルシート | Transitional | | | |
| | lang | 言語コード | Transitional | | | |
| | dir | 文字表記の方向 | Transitional | | | |
| ul | | 番号なしリスト | Strict | | | 64 |
| | type | マークの種類 | Transitional | 非推奨 | | 65 |
| | compact | 詰めて表示 | Transitional | 非推奨 | | |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |
| var | | 変数・引数 | Strict | | | 39 |
| | class | クラス名 | Strict | | | |
| | id | ID名 | Strict | | | |
| | title | 補足情報 | Strict | | | |
| | style | スタイルシート | Strict | | | |
| | lang | 言語コード | Strict | | | |
| | dir | 文字表記の方向 | Strict | | | |

※イベント属性は省略しています。イベントについては、JavaScriptパートを参照してください。

# HTML4.01 対応状況一覧

| 要素名 | 説明 | Windows 版 | | | | | | | | | Macintosh 版 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | IE6 | IE5.5 | IE5.0 | IE4 | Firefox | N7/6 | N4 | O7 | O6 | Safari | IE5 | IE4 |
| a | リンク(アンカー) | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| abbr | 略語 | × | × | × | × | ○ | ○ | × | ○ | ○ | ○ | ○ | × |
| acronym | 頭字語 | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| address | 連絡先 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| applet | JAVA アプレット | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| area | イメージマップの領域指定 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| b | 太字 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| base | 基準 URL | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| basefont | 基準フォント・色・サイズ | ○ | ○ | ○ | ○ | × | × | ○ | × | × | × | ○ | ○ |
| bdo | 文字表記の方向 | ○ | ○ | ○ | × | ○ | ○ | × | ○ | × | × | × | × |
| big | 大きな文字 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| blockquote | ブロックレベルの引用文 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| body | 文書本体 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| br | 改行 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| button | ボタン | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | △ |
| caption | 表のタイトル | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| center | 中央揃え | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| cite | 出典・参照先 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| code | ソースコード | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| col | 表の縦列 | ○ | ○ | ○ | ○ | △ | △ | × | ○ | △ | × | △ | △ |
| colgroup | 表の縦列のグループ化 | ○ | ○ | ○ | ○ | △ | △ | × | ○ | × | × | △ | △ |
| dd | dl 要素の説明部分 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| del | 削除部分 | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| dfn | 定義対象の用語 | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| dir | ディレクトリ・リスト | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| div | ブロックレベルの範囲の設定 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| dl | 用語と説明のリスト | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| dt | dl 要素の用語部分 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| em | 強調 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| fieldset | フォームの内容のグループ化 | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| font | フォントの種類・色・サイズ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| form | 入力フォーム | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| frame | フレームの内容指定 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| frameset | フレームの定義 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| h1～h6 | 見出し | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| head | 文書情報 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| hr | 横罫線 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| html | 最上位要素 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

| 要素名 | 説明 | Windows版 | | | | | | | | | Macintosh版 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | IE6 | IE5.5 | IE5.0 | IE4 | Firefox | N7/6 | N4 | O7 | O6 | Safari | IE5 | IE4 |
| i | イタリック | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| iframe | インラインフレーム | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| img | 画像 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| input | 入力·選択用部品とボタン | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| ins | 追加部分 | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | △ |
| isindex | 検索用入力フィールド | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| kbd | 入力テキスト | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| label | フォームの部品のラベル | △ | △ | △ | △ | ○ | ○ | × | ○ | × | × | ○ | ○ |
| legend | fieldset 要素のラベル | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | △ | ○ | ○ | ○ |
| li | リスト項目 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| link | 関連ファイル | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| map | イメージマップの定義 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| menu | メニューリスト | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| meta | メタ情報 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| noframes | フレームが表示できない環境用 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| noscript | スクリプトが動作しない環境用 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| object | オブジェクト | △ | △ | △ | △ | ○ | ○ | × | ○ | ○ | ○ | △ | × |
| ol | 番号付きのリスト | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| optgroup | メニュー選択肢のグループ化 | △ | × | × | × | △ | △ | × | △ | × | △ | ○ | × |
| option | メニューの選択肢 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| p | 段落(パラグラフ) | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| param | パラメータ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| pre | 整形済みテキスト | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| q | インラインの引用文 | △ | △ | △ | △ | ○ | ○ | × | ○ | ○ | ○ | ○ | △ |
| s | 取消線 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| samp | 出力サンプル | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| script | スクリプト | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| select | メニュー | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| small | 小さな文字 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| span | インラインの範囲の設定 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| strike | 取消線 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| strong | より強い強調 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| style | スタイルシート | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| sub | 下付き文字 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| sup | 上付き文字 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| table | 表 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| tbody | 表本体 | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| td | データセル | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| textarea | 複数行の入力フィールド | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| tfoot | 表フッタ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| th | 見出しセル | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

| 要素名 | 説明 | Windows版 | | | | | | | | | Macintosh版 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | IE6 | IE5.5 | IE5.0 | IE4 | Firefox | N7/6 | N4 | O7 | O6 | Safari | IE5 | IE4 |
| thead | 表ヘッダ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| title | 文書タイトル | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| tr | 表の横一列 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| tt | 等幅フォント | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ |
| u | 下線 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| ul | 番号無しリスト | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| var | 変数・引数 | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

※各要素の対応状況を、○△×の3段階でおおまかに示しました。
環境や細かいバージョンの違いなどによっては、結果が異なる場合もありますので、注意してください。

# i モード端末で利用できる要素・属性一覧

| 要素名 | 属性名 |
|---|---|
| **ページの基礎となる内容** | |
| HTML | |
| HEAD | |
| BODY | |
| TITLE | |
| BASE | href |
| <!-- --> | |
| **テキスト** | |
| H1～H6 | align |
| P | align |
| BLOCKQUOTE | |
| PLAINTEXT | |
| DIV | align |
| **スタイルとレイアウト** | |
| BR | clear |
| PRE | |
| CENTER | |
| HR | align・size・width |
| **リンク** | |
| A | name・href・accesskey |
| **リスト** | |
| UL | |
| OL | |
| LI | |
| DL | |
| DT | |
| DD | |
| DIR | |
| MENU | |
| **画像** | |
| IMG | src・align・width・height・hspace・vspace・alt |
| **フォーム** | |
| FORM | action・method |
| INPUT | type・name・size・maxlength・value・checked・accesskey |
| TEXTAREA | name・rows・cols・accesskey |
| SELECT | name・size |
| OPTION | selected |

※i モード対応のHTMLには、1.0から5.0までのバージョンがありますが、ここでは全機種で動作可能な「i モード対応HTML1.0」の要素と属性を紹介します。

※より詳しい内容については、「NTT DoCoMo Net」のホームページを参照してください。

**「DoCoMo Net - 作ろう i モードコンテンツ」**

(http://www.nttdocomo.co.jp/p_s/imode/make/)

# HTML4.01で定義されている特別な文字

「テキストの種類」の「特別な文字を表示させる」(P.41)では、日本語環境でも使用できる特別な文字について説明しましたが、HTML4.01ではそれ以外にも多くの文字が定義されています。しかし、実際に表示できる文字は、使用しているOSやブラウザのバージョンなどによって異なります。特に日本語環境で利用している場合には、ほとんどが表示できない場合もあります。ここでは、その中でも多くの文字を表示可能なWindows版のInternet Explorer 6.0での表示例を示します。

「特別な文字」は、HTML4.01では文字参照として定義されており、次の3種類に分類されています。

- ISO 8859-1 (欧米:Latin 1)
- ギリシャ文字・シンボル・数学記号
- その他の特別な文字

ここに定義されている文字を表示させる場合、「&キーワード;」という形式と、「&#番号;」という2種類の形式を利用することができます。キーワードについては、大文字と小文字は別の文字として扱われますので、注意してください。なお、キーワード形式では表示されないものでも、番号形式で指定すると表示される場合があります。

## ISO 8859-1 (Latin 1)

| | キーワード | #番号 | | キーワード | #番号 | | キーワード | #番号 |
|---|---|---|---|---|---|---|---|---|
| |   |   | ¡ | ¡ | ¡ | ¢ | ¢ | ¢ |
| £ | £ | £ | ¤ | ¤ | ¤ | ¥ | ¥ | ¥ |
| ¦ | ¦ | ¦ | § | § | § | ¨ | ¨ | ¨ |
| © | © | © | ª | ª | ª | « | « | « |
| ¬ | ¬ | ¬ | |  |  | ® | ® | ® |
| ¯ | ¯ | ¯ | ° | ° | ° | ± | ± | ± |
| ² | ² | ² | ³ | ³ | ³ | ´ | ´ | ´ |
| µ | µ | µ | ¶ | ¶ | ¶ | · | · | · |
| ¸ | ¸ | ¸ | ¹ | ¹ | ¹ | º | º | º |
| » | » | » | ¼ | ¼ | ¼ | ½ | ½ | ½ |
| ¾ | ¾ | ¾ | ¿ | ¿ | ¿ | À | À | À |
| Á | Á | Á | Â | Â | Â | Ã | Ã | Ã |
| Ä | Ä | Ä | Å | Å | Å | Æ | Æ | Æ |
| Ç | Ç | Ç | È | È | È | É | É | É |
| Ê | Ê | Ê | Ë | Ë | Ë | Ì | Ì | Ì |
| Í | Í | Í | Î | Î | Î | Ï | Ï | Ï |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Ð | Ð | Ð | Ñ | Ñ | Ñ | Ò | Ò | Ò |
| Ó | Ó | Ó | Ô | Ô | Ô | Õ | Õ | Õ |
| Ö | Ö | Ö | × | × | × | Ø | Ø | Ø |
| Ù | Ù | Ù | Ú | Ú | Ú | Û | Û | Û |
| Ü | Ü | Ü | Ý | Ý | Ý | Þ | Þ | Þ |
| ß | ß | ß | à | à | à | á | á | á |
| â | â | â | ã | ã | ã | ä | ä | ä |
| å | å | å | æ | æ | æ | ç | ç | ç |
| è | è | è | é | é | é | ê | ê | ê |
| ë | ë | ë | ì | ì | ì | í | í | í |
| î | î | î | ï | ï | ï | ð | ð | ð |
| ñ | ñ | ñ | ò | ò | ò | ó | ó | ó |
| ô | ô | ô | õ | õ | õ | ö | ö | ö |
| ÷ | ÷ | ÷ | ø | ø | ø | ù | ù | ù |
| ú | ú | ú | û | û | û | ü | ü | ü |
| ý | ý | ý | þ | þ | þ | ÿ | ÿ | ÿ |

## ギリシャ文字・シンボル・数学記号

| | キーワード | #番号 | | キーワード | #番号 | | キーワード | #番号 |
|---|---|---|---|---|---|---|---|---|
| ƒ | ƒ | ƒ | Α | Α | Α | Β | Β | Β |
| Γ | Γ | Γ | Δ | Δ | Δ | Ε | Ε | Ε |
| Ζ | Ζ | Ζ | Η | Η | Η | Θ | Θ | Θ |
| Ι | Ι | Ι | Κ | Κ | Κ | Λ | Λ | Λ |
| Μ | Μ | Μ | Ν | Ν | Ν | Ξ | Ξ | Ξ |
| Ο | Ο | Ο | Π | Π | Π | Ρ | Ρ | Ρ |
| Σ | Σ | Σ | Τ | Τ | Τ | Υ | Υ | Υ |
| Φ | Φ | Φ | Χ | Χ | Χ | Ψ | Ψ | Ψ |
| Ω | Ω | Ω | α | α | α | β | β | β |
| γ | γ | γ | δ | δ | δ | ε | ε | ε |
| ζ | ζ | ζ | η | η | η | θ | θ | θ |
| ι | ι | ι | κ | κ | κ | λ | λ | λ |
| μ | μ | μ | ν | ν | ν | ξ | ξ | ξ |
| ο | ο | ο | π | π | π | ρ | ρ | ρ |
| ς | ς | ς | σ | σ | σ | τ | τ | τ |
| υ | υ | υ | φ | φ | φ | χ | χ | χ |
| ψ | ψ | ψ | ω | ω | amp;969; | ϑ | ϑ | ϑ |
| | ϒ | ϒ | | &piv | ϖ | • | • | • |
| … | … | … | ′ | ′ | ′ | ″ | ″ | ″ |
| ‾ | ‾ | ‾ | ⁄ | ⁄ | ⁄ | | ℘ | ℘ |
| | ℑ | ℑ | | ℜ | ℜ | ™ | ™ | ™ |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | ℵ | ℵ | ← | ← | ← | ↑ | ↑ | ↑ |
| → | → | → | ↓ | ↓ | ↓ | ↔ | ↔ | ↔ |
| | ↵ | ↵ | ← | ⇐ | ⇐ | ↑ | ⇑ | ⇑ |
| ⇒ | ⇒ | ⇒ | | ⇓ | ⇓ | ⇔ | ⇔ | ⇔ |
| ∀ | ∀ | ∀ | ∂ | ∂ | ∂ | ∃ | ∃ | ∃ |
| | ∅ | ∅ | ∇ | ∇ | ∇ | ∈ | ∈ | ∈ |
| | ∉ | ∉ | ∋ | ∋ | ∋ | ∏ | ∏ | ∏ |
| Σ | ∑ | ∑ | − | − | − | | ∗ | ∗ |
| √ | √ | √ | ∝ | ∝ | ∝ | ∞ | ∞ | ∞ |
| ∠ | ∠ | ∠ | ∧ | ∧ | ∧ | ∨ | ∨ | ∨ |
| ∩ | ∩ | ∩ | ∪ | ∪ | ∪ | ∫ | ∫ | ∫ |
| ∴ | ∴ | ∴ | ∼ | ∼ | ∼ | | ≅ | ≅ |
| ≈ | ≈ | ≈ | ≠ | ≠ | ≠ | ≡ | ≡ | ≡ |
| ≤ | ≤ | ≤ | ≥ | ≥ | ≥ | ⊂ | ⊂ | ⊂ |
| ⊃ | ⊃ | ⊃ | | ⊄ | ⊄ | ⊆ | ⊆ | ⊆ |
| ⊇ | ⊇ | ⊇ | ⊕ | ⊕ | ⊕ | | ⊗ | ⊗ |
| ⊥ | ⊥ | ⊥ | | ⋅ | ⋅ | | ⌈ | ⌈ |
| | ⌉ | ⌉ | | ⌊ | ⌊ | | ⌋ | ⌋ |
| | ⟨ | 〈 | | ⟩ | 〉 | ◊ | ◊ | ◊ |
| ♠ | ♠ | ♠ | ♣ | ♣ | ♣ | ♥ | ♥ | ♥ |
| ♦ | ♦ | ♦ | | | | | | |

## その他の特別な文字

| | キーワード | #番号 | | キーワード | #番号 | | キーワード | #番号 |
|---|---|---|---|---|---|---|---|---|
| " | " | " | & | & | & | < | < | < |
| > | > | > | Œ | Œ | Œ | œ | œ | œ |
| Š | Š | Š | š | š | š | Ÿ | Ÿ | Ÿ |
| ˆ | ˆ | ˆ | ˜ | ˜ | ˜ | |   |   |
| |   |   | |   |   | | ‌ | ‌ |
| | ‍ | ‍ | | ‎ | ‎ | | ‏ | ‏ |
| – | – | – | — | — | — | ‘ | ‘ | ‘ |
| ’ | ’ | ’ | ‚ | ‚ | ‚ | “ | “ | “ |
| ” | ” | ” | „ | „ | „ | † | † | † |
| ‡ | ‡ | ‡ | ‰ | ‰ | ‰ | ‹ | ‹ | ‹ |
| › | › | › | € | € | € | | | |

※このサンプルでは、Windows XP上でInternet Explorer 6.0を使い、もっとも一般的と思われる設定(文字コードは「シフトJIS」・フォントは「MS ゴシック」)で、番号形式で文字を指定して表示させました。表示されない、または文字化けしてしまう文字に関しては、空欄にしてあります。
詳細な使用方法は、「テキストの種類」の「特別な文字を表示させる」(P.41)を参照してください。

詳解
HTML&CSS&JavaScript辞典 改訂版

The revised edition of the detailed explanation dictionary about HyperText Markup Language, Cascading Style Sheets, and JavaScript.

# CSS

# スタイルシートについて

HTMLは基本的に文書の構成要素の意味や役割を示しますが、その構成要素を具体的にどのように表示させるかを指定するのがスタイルシートです。スタイルシートを導入すると、HTMLでは不可能な多くの表現が可能になるだけでなく、HTMLがシンプルになって容量が減り、ページの更新や修正などのメンテナンスも大幅に楽になります。また、表示に関する指定をHTMLから取り除くことによって、そのページがより多くの環境で利用できるアクセシブルなものにもなります。

しかし、スタイルシート自体はそれほど難しくはないのですが、各ブラウザのスタイルシートへの対応状況には、注意する必要があります。特に古いブラウザの中には、正しく表示できないものや、間違った解釈で表示するものが多くあります。本書では、各解説ページやリファレンスでブラウザの対応状況を掲載していますので、参考にしてください。

なお、「スタイルシート」といっても実際には何種類かの言語が存在するのですが、本書でスタイルシートという場合には、現在もっとも広く利用が可能なCSS(Cascading Style Sheet)のことを指しています。そして本書では、その中から現在、実際に利用可能なものを中心に解説しています。

CSSについて

# 基本的な書き方

CSSは、基本的に次のような書式で記述されます。

```
セレクタ { プロパティ:値 }
```

セレクタとは、どの要素に対してスタイルを適用させるかを指定する部分です。この部分でスタイルの適用対象を示し、それに続く「{～}」の中に適用させたいスタイルを記述します。プロパティとは、セレクタで指定した要素に適用するスタイルの種類を示す部分です。色を表す「color」や、フォントサイズを表す「font-size」などがこれにあたります。これに続けて「:」記号と値を記述することでスタイルを設定することができます。スタイルは、「;」で区切って複数指定することができます。

・h1要素のフォントサイズを24ポイントに設定した例

```
h1 { font-size: 24pt }
```

・h1要素のフォントサイズを24ポイントに、色を青に設定した例

```
h1 { font-size: 24pt; color: #0000ff }
```

指定するスタイルが多い場合には、次のような書き方もできます。この場合、スタイルとスタイルの間を「;」で区切ることを忘れないようにしてください。

```
h1 {
  font-size: 24pt;
  color: blue;
      ∫
}
```

## 単位について

CSSで大きさを指定する場合には、数値に次の単位を付けて示すことができます。これらは、絶対的な値を示す単位と、相対的な値を示す単位のふたつに分類することができます。具体的な大きさについては、巻末付録の「Webサイズチャート」を参照してください。

### 絶対的な値を示す単位

| | |
|---|---|
| in | インチ（1インチ=2.54cm） |
| cm | センチメートル |
| mm | ミリメートル |
| pt | ポイント（1ポイント=1/72インチ） |
| pc | パイカ（1パイカ=12ポイント） |

### 相対的な値を示す単位

| | |
|---|---|
| em | その範囲で有効なフォントの高さを1とする単位 |
| ex | その範囲で有効なフォントの小文字の「x」の高さを1とする単位 |
| px | 1ピクセルを1とする単位 |
| % | 他の基準となる大きさに対する割合（基準はそれぞれ異なります） |

※値が0の場合は、単位を省略することができます。

# 色の指定方法

CSSで色を指定するには、以下の5つの方法があります。具体的な色とその値については、巻末付録の「カラーチャート1～3」を参照してください。

## 16進数で指定する1（#RRGGBB形式）

HTMLの場合と同様な指定方法です。「#」記号に続けて、RGB(Red, Green, Blue)の各値を2桁ずつの16進数で示します。

たとえば、赤を指定する場合には「#ff0000」となります。

## 16進数で指定する2（#RGB形式）

この指定方法は上記の方法に似ていますが、RGBの各値をそれぞれ1桁ずつの16進数で示します。この値は、指定したRGBの各1桁の値をふたつ続けて並べた数値として解釈されて表示されます。

たとえば、「#f36」と指定した場合には、「#ff3366」を指定したのと同じ結果になります。赤を指定する場合には「#f00」となります。

## 10進数で指定する（rgb(n,n,n)形式）

この指定方法では、rgbに続く()の中にRとGとBの10進数の値をそれぞれ「,」で区切って示します。RGBの各値は0～255の範囲で指定することができます。

たとえば、赤を指定する場合には「rgb(255,0,0)」となります。

## %で指定する（rgb(n%,n%,n%)形式）

この指定方法では、rgbに続く()の中にRとGとBのパーセントの値をそれぞれ「,」で区切って示します。RGBの各値は0%～100%の範囲で指定することができます。

たとえば、赤を指定する場合には「rgb(100%,0%,0%)」となります。

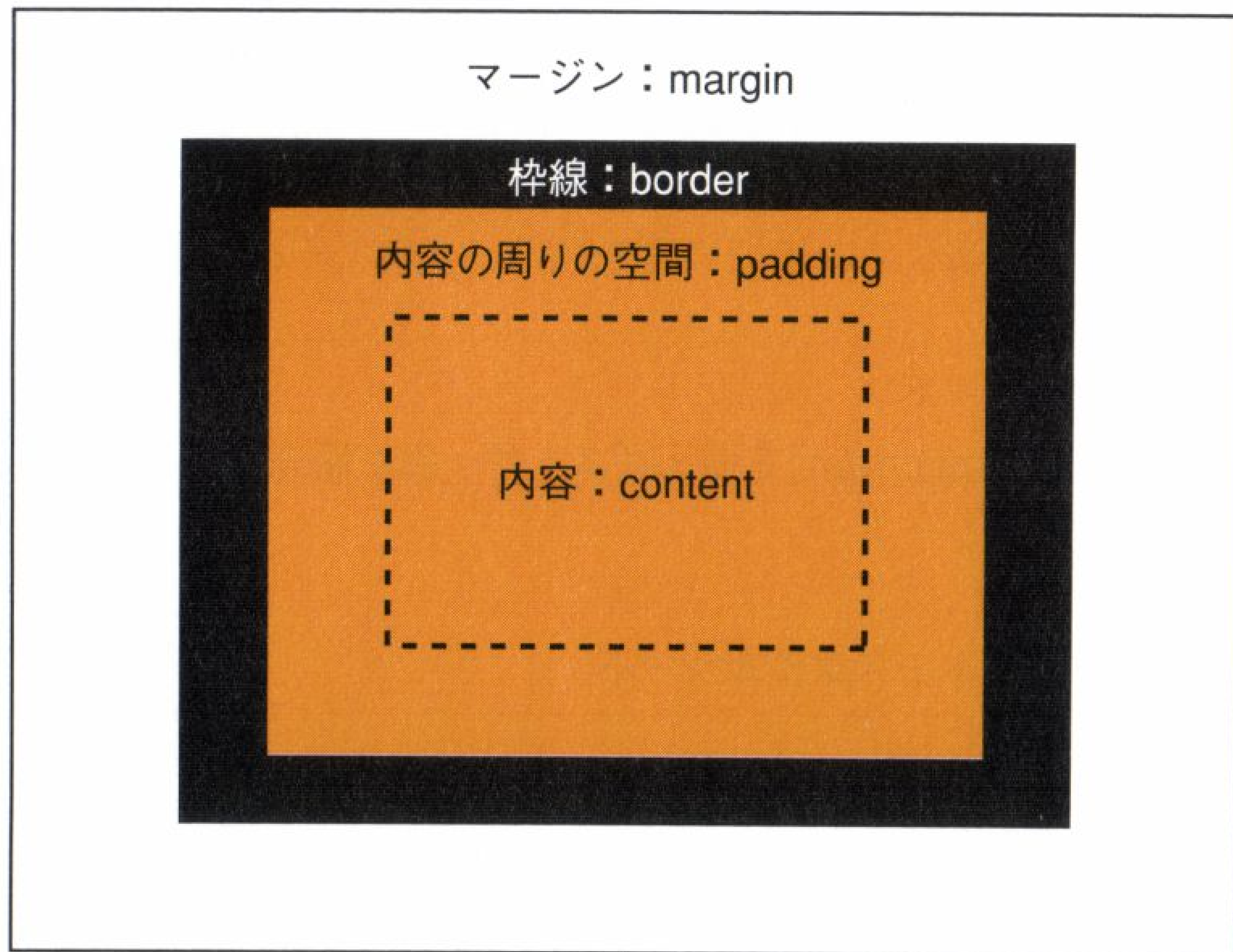
## 色の名前で指定する

HTML4.01で定義されている16色を名前で直接指定することができます。また、システムで利用されている色を指定できるキーワードも用意されているのですが、本書では省略します。

# ボックスについて

各要素は、ボックスと呼ばれる四角い領域を生成します。この領域は、次の４つの部分から構成されています。

**ボックスの構造**




・内容（content）

テキストや画像などの、要素の内容が表示される領域です。この領域の幅と高さは、width プロパティと height プロパティで指定することができます。

・内容の周りの空間（padding）

内容が表示される部分と枠線の間の空間部分の領域です。要素に対して指定された背景は、この領域にも描画されます。

・枠線（border）

要素の周りに表示させることのできる枠の部分を示します。

・マージン（margin）

ボックスの一番外側のマージン領域を示します。要素に対して指定された背景は、ここには表示されず、背景は常に透明となります。

# スタイルの優先順位

スタイルシートは、その文書を制作した人だけが設定できるというものではありません。ブラウザによっては、ユーザーが任意のスタイルシートを適用できるようになっているものもあります。また、ブラウザはデフォルトのスタイルシート持っていて、最初にそれを適用することになっています。つまり、ひとつの文書に対して、「制作者」「ユーザー」「ブラウザ」の三者から、同時にスタイルシートが適用される可能性があるわけです。そして、スタイルシートが部分的に競合する可能性も出てきます。

そのようにスタイルが競合する場合の優先順は、次のようになっています。

- 制作者のスタイルシート
  - ⬆優先（ただし「!important」を使えばユーザー側が優先）
- ユーザーのスタイルシート
  - ⬆優先
- ブラウザのデフォルト・スタイルシート

通常は「ユーザー」のスタイルシートよりも「制作者」のスタイルシートが優先されるのですが、「!important」というキーワードを使用することで、これを逆転させることもできます。「!important」は、次のように優先させたい「プロパティ：値」の後に指定します。

```
p { font-size: 18pt !important }
```

このキーワードは、「制作者」のスタイルシートでも使用することができるため、「!important」自体も競合する可能性があります。しかし、この場合も「ユーザー」のスタイルシートが優先される仕様になっています。

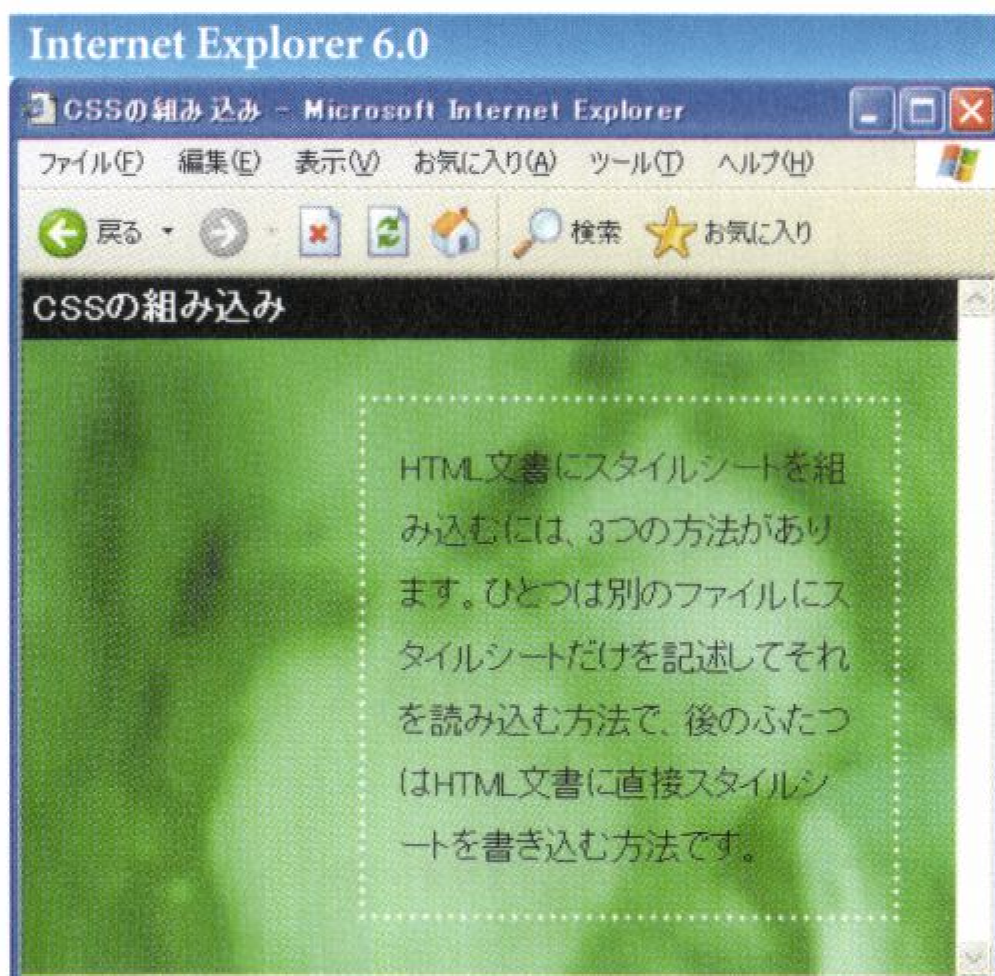
また、同じスタイルシート内でも競合が起こることがあります。その場合でも「!important」を使ってスタイルを優先させることができますが、一般的にはより細かい部分で指定しているスタイルが優先されます。たとえば、要素に対して指定したスタイルよりはクラスに対して指定したスタイル、クラスに対して指定したスタイルよりはidで指定したスタイルが優先されます。もし、それでも競合してしまう場合には、より後に指定されたスタイルが優先されます。

# CSSの書かれたファイルを読み込む

**<link rel="stylesheet" href="*URL*" type="text/css">**

| | |
|---|---|
| URL | 読み込むCSSファイルのURL |

Internet Explorer 6.0




Firefox 1.0




**スタイルシートの読み込み**

HTML文書

```
<!DOCTYPE ~ >
<html>
<head>
<link ~ >
</head>
<body>
</body>
</html>
```

スタイルシート

```
body {
  margin: ~
  color:  ~
  background: ~
}
    ・
    ・
    ・
    ・
```

**1つのスタイルシートを複数のHTML文書へ**

HTML文書

<link ~ >
<link ~ >
<link ~ >

スタイルシート

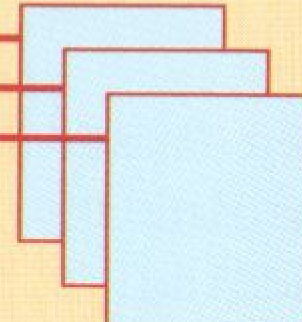


**複数のスタイルシートを1つのHTML文書へ**

HTML文書

<link ~ >
<link ~ >
<link ~ >

スタイルシート



スタイルシートのみを記述した別のファイル（拡張子は通常「.css」）を用意して、それをHTML文書に読み込んで適用させたい場合には、link要素を使用します。
この要素は、必ず<head>〜</head>の範囲内に配置するようにしてください。複数のファイルを読み込みたい場合には、この要素を必要な数だけ配置することができます。
また、複数のHTML文書で共通するCSSをまとめて別ファイルにしておき、それを読み込んで利用するようにしておけば、CSSファイルを修正するだけで一度に複数のHTML文書の表示方法を変更できるようになります。

## Sample

**【HTML】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>CSSの組み込み</title>
<link rel="stylesheet" href="default.css" type="text/css">
</head>
<body>
<h1>CSSの組み込み</h1>
<p>
HTML文書にスタイルシートを組み込むには、3つの方法があります。
ひとつは別のファイルにスタイルシートだけを記述してそれを読み込む
方法で、後のふたつはHTML文書に直接スタイルシートを書き込む方法です。
</p>
</body>
</html>
```

**【CSS（default.cssの内容）】**

```
body {
  margin: 0;
  color: #000000;
  background: #00cc99 url(back.jpg)
}
h1 {
  font-size: medium;
  margin: 0;
  padding: 0.3em;
  color: #ffffff;
  background-color: #000000
}
```

```
p {
  line-height: 1.8;
  margin: 1.6em 1.6em 1em 36%;
  padding: 1em;
  border: dotted 3px #ffffff
}
```

HTML：「文書情報」の「関連する他のページを示す」(P.22)
HTML：コラム「ページ同士の関係を表す値」(P.23)

TIPS

### Netscape Navigator 4.XにだけCSSを適用させない方法

link要素でCSSの書かれた別ファイルを読み込む場合、media属性を使用して、そのスタイルシートが想定している出力対象を指定することができます。指定可能な値は以下のもので、「,」で区切って複数指定することもできます。

| 値 | 概要 |
|---|---|
| all | すべて |
| screen | 一般のコンピュータの画面（デフォルト） |
| tty | 文字幅が固定の機器 |
| tv | テレビ |
| projection | プロジェクタ |
| handheld | 携帯用端末 |
| print | プリンタ |
| braille | 点字ディスプレイ |
| embossed | 点字プリンタ |
| aural | 音声による出力 |

ところで、Netscape Navigator 4.Xは、このmedia属性の値として「screen」以外を指定すると、スタイルシートを読み込まないようになっています。したがって、Netscape Navigator 4.Xにだけスタイルシートを適用したくない場合には、「media="all"」や「media="screen,tv"」などのように指定しておけばOKです。

```
<link rel="stylesheet" href="base.css" type="text/css"
 media="screen,tv">
```

CSSを組み込む

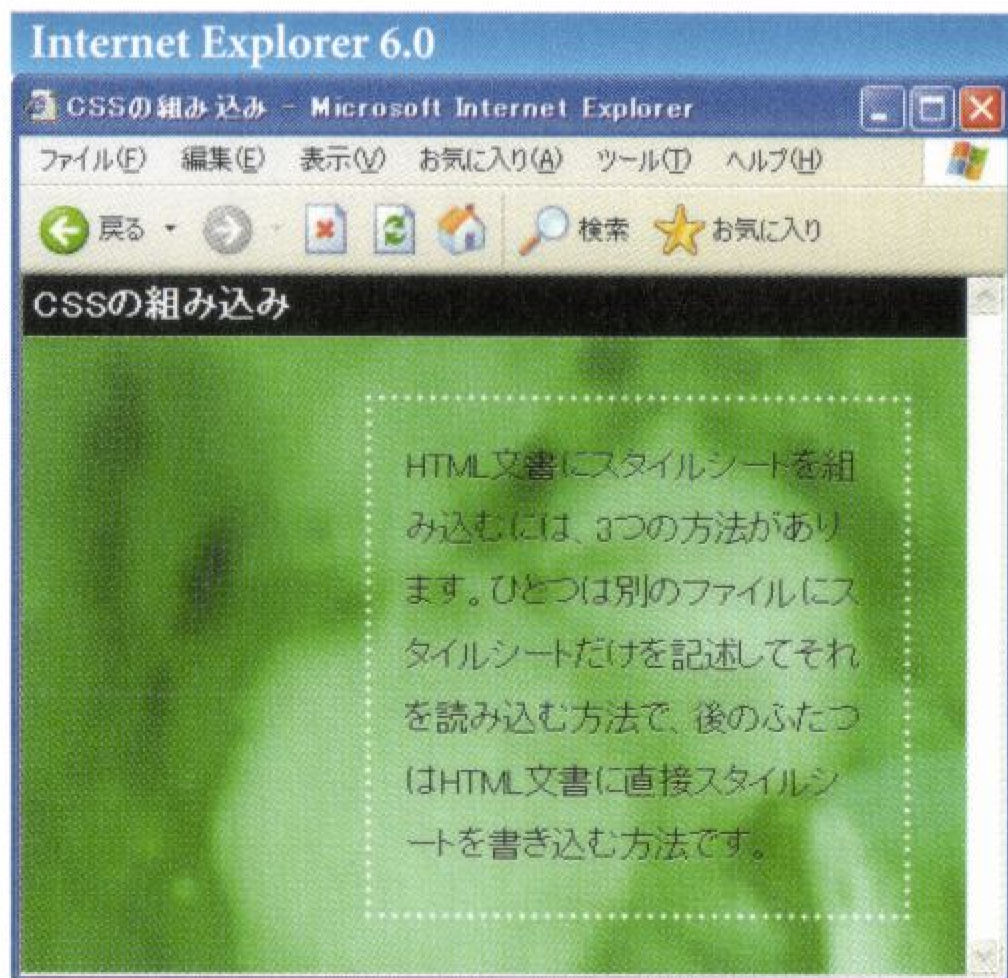
IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# style要素の内容として組み込む

## <style type="text/css">～</style>

Internet Explorer 6.0




Firefox 1.0




HTML文書の中にCSSを書き込む場合には、style要素を使用します。
この要素は、必ず<head>～</head>の範囲内に配置するようにしてください。<style>～</style>の範囲には、直接CSSを書き込むことができます。この時、スタイルシートに未対応のブラウザがCSSのソースをテキストとして表示してしまうことを避けるために、CSS全体をHTMLでのコメント(<!--～-->)にしておくことができます。

### Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>CSSの組み込み</title>
<style type="text/css">
<!--
body {
  margin: 0;
  color: #000000;
  background: #00cc99 url(back.jpg)
}
h1 {
  font-size: medium;
  margin: 0;
  padding: 0.3em;
```

CSSを組み込む

```
  color: #ffffff;
  background-color: #000000
}
p {
  line-height: 1.8;
  margin: 1.6em 1.6em 1em 36%;
  padding: 1em;
  border: dotted 3px #ffffff
}
-->
</style>
</head>
<body>
<h1>CSSの組み込み</h1>
<p>
HTML文書にスタイルシートを組み込むには、3つの方法があります。
ひとつは別のファイルにスタイルシートだけを記述してそれを読み込む
方法で、後のふたつはHTML文書に直接スタイルシートを書き込む方法です。
</p>
</body>
</html>
```

TIPS 「Netscape Navigator 4.XにだけCSSを適用させない方法」(P.185)

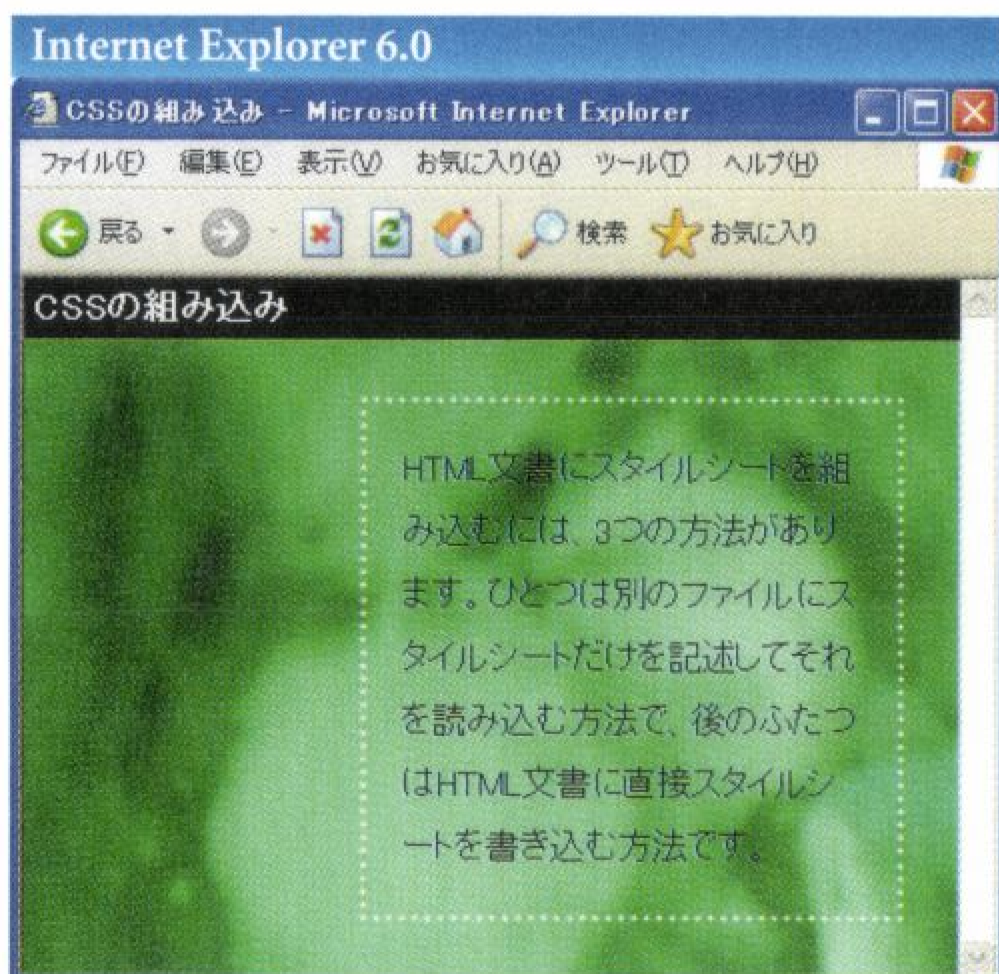
CSS

# 任意の要素にstyle属性の値として組み込む

<要素名 style="スタイルシート">
<要素名 style="スタイルシート; スタイルシート; …">

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

Internet Explorer 6.0




Firefox 1.0




HTML文書中の任意のタグにstyle属性を追加して、その要素だけに有効なCSSを書き込むことができます。
複数の指定をしたい場合には、それぞれを「;」で区切って指定します。この方法を利用する場合は、そこに書かれているスタイルシート言語(CSS以外もありえます)を特定する手段がありませんので、meta要素を使用して必ずデフォルトのスタイルシート言語を宣言しておくようにしてください。
なお、style属性は、HTMLの新しい標準仕様(XHTML1.1)では非推奨とされています。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" ↵
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html; ↵
 charset=Shift_JIS">
<meta http-equiv="Content-Style-Type" content="text/css">
<title>CSSの組み込み</title>
</head>
<body style="margin:0; color:#000000; background:#00cc99 ↵
 url(back.jpg)">
<h1 style="font-size: medium; margin: 0; padding: 0.3em; color: ↵
#ffffff; background-color: #000000">
CSSの組み込み
</h1>
```

```
<p style="line-height: 1.8; margin: 1.6em 1.6em 1em 36%; padding: 1em; border: dotted 3px #ffffff">
HTML文書にスタイルシートを組み込むには、3つの方法があります。
ひとつは別のファイルにスタイルシートだけを記述してそれを読み込む
方法で、後のふたつはHTML文書に直接スタイルシートを書き込む方法です。
</p>
</body>
</html>
```

 HTML ：「文書情報」の「スタイルシートやスクリプトの言語を示す」(P.21)

## コラム

### スタイルシートは別ファイルにするのがベスト

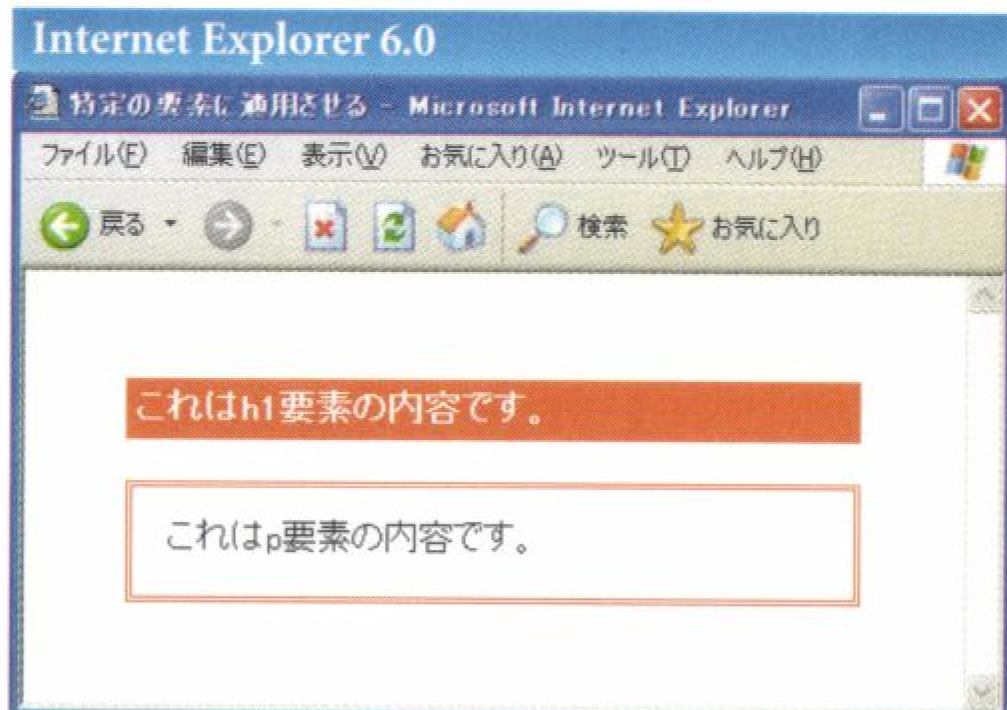
左ページのstyle属性の解説でも触れましたが、XHTML1.1ではstyle属性は非推奨とされています。また、style要素を使う方法に関しては、XHTMLでは特定の文字やHTMLのコメントが使えないなど、色々と不都合が生じます。将来HTMLをXHTMLに変更する可能性がある場合は、スタイルシートを別ファイルで用意して、それを読み込むようにしておいたほうがよいでしょう。また、そのようにすると、特定のブラウザにだけスタイルシート適用させないようにすることも可能になります(P.185のTIPS「Netscape Navigator 4.XにだけCSSを適用させない方法」参照)。

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

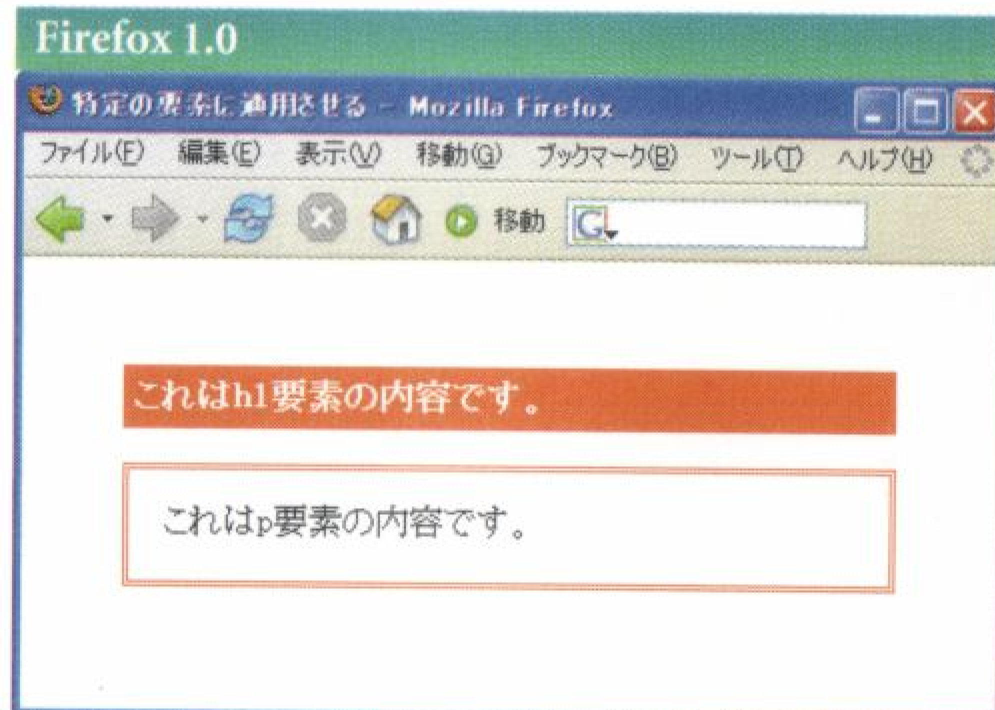
# 特定の要素に適用させる

## 要素名 { ~ }

Internet Explorer 6.0




Firefox 1.0




指定した要素に対して、スタイルを適用させたい場合の書式です。

## Sample

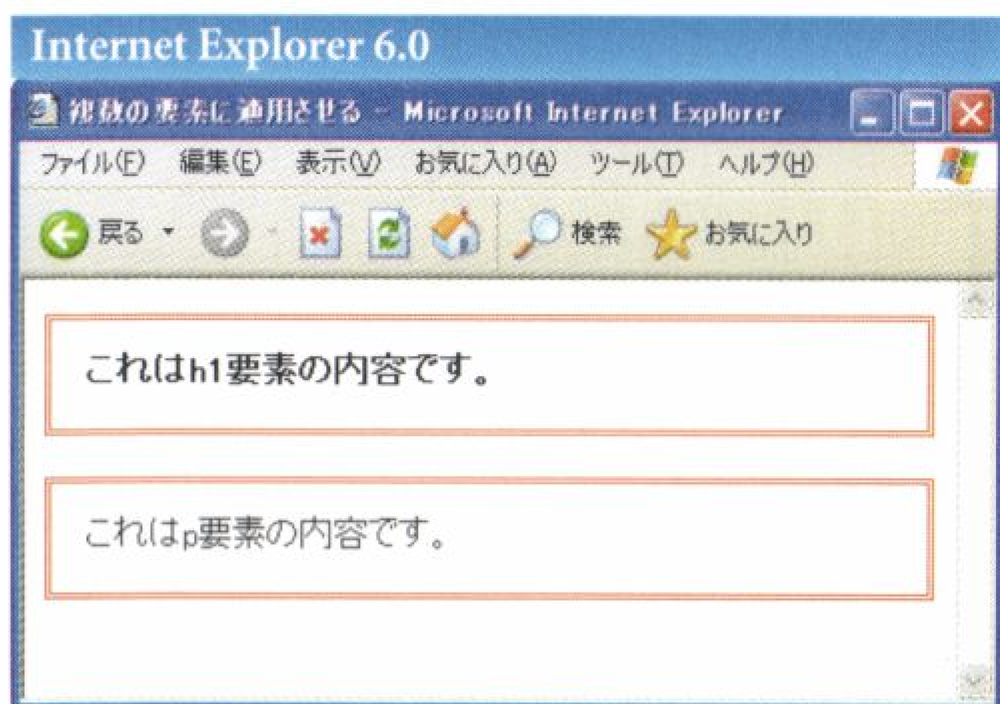
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>特定の要素に適用させる</title>
<style type="text/css">
<!--
body { margin: 3em }
h1 {
  font-size: medium;           /* 文字サイズを標準に */
  padding: 0.3em;
  color: #ffffff;              /* 文字色を白に */
  background: #ff6600          /* 背景色をオレンジに */
}
p {
  padding: 1em;
  border: double 3px #ff6600   /* オレンジの2重線で囲む */
}
-->
</style>
</head>
<body>
<h1>これはh1要素の内容です。</h1>
<p>これはp要素の内容です。</p>
</body>
</html>
```

# 複数の要素に適用させる

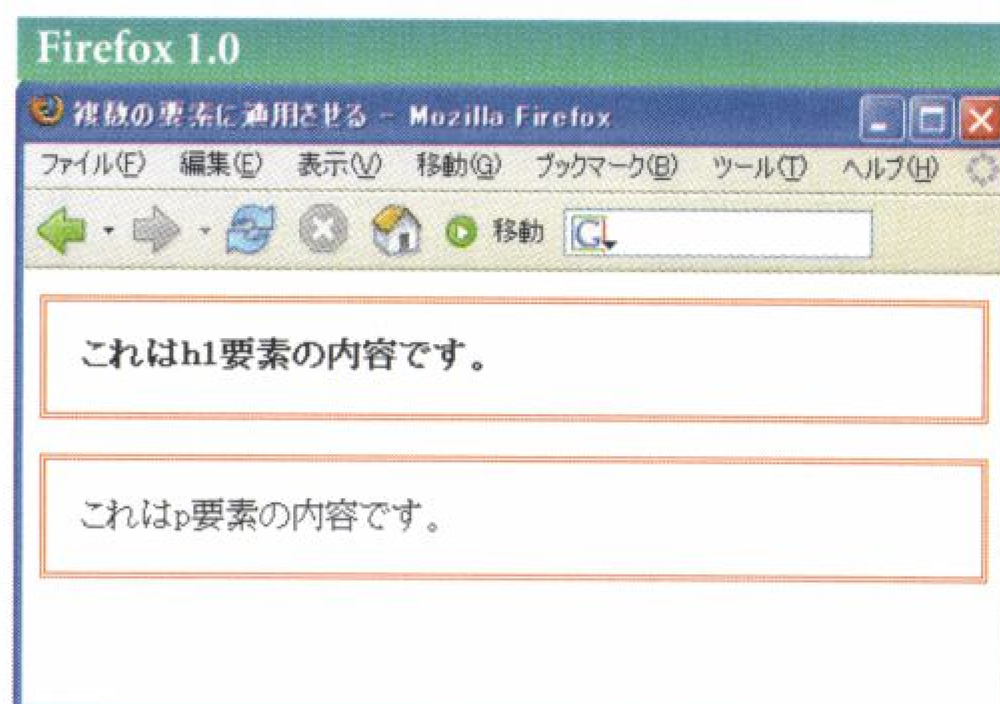
**要素名, 要素名, 要素名, … { ~ }**

※「要素名」の部分には、「#ID名」や「.クラス名」も指定できます。

Internet Explorer 6.0



Firefox 1.0



指定した複数の要素に対して、同じスタイルを適用させたい場合の書式です。要素の名前だけでなく、「#ID名」や「.クラス名」を指定することもできます。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>複数の要素に適用させる</title>
<style type="text/css">
<!--
h1, p {
  font-size: medium;            /* 文字サイズを標準に */
  padding: 1em;
  border: double 3px #ff6600   /* オレンジの2重線で囲む */
}
-->
</style>
</head>
<body>
<h1>これはh1要素の内容です。</h1>
<p>これはp要素の内容です。</p>
</body>
</html>
```

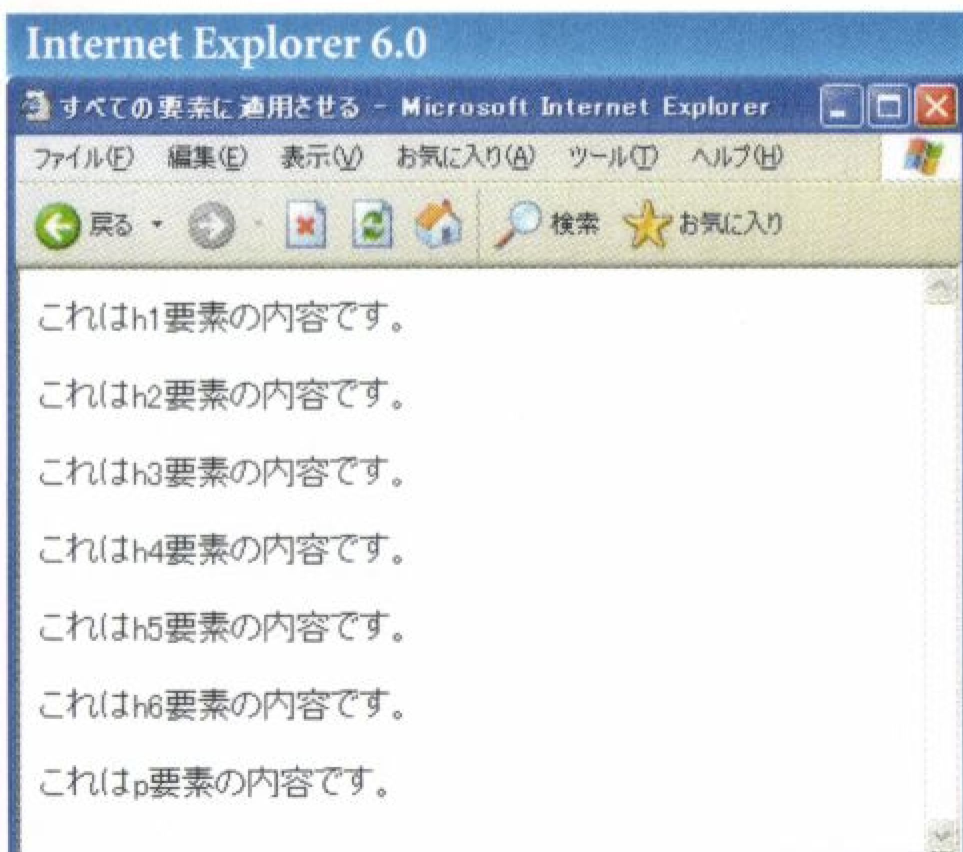

要素名：「CSSを適用する対象」の「IDやクラスを指定した要素に適用させる」(P.193)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X Opera7 Opera6 Safari IE5-mac IE4-mac

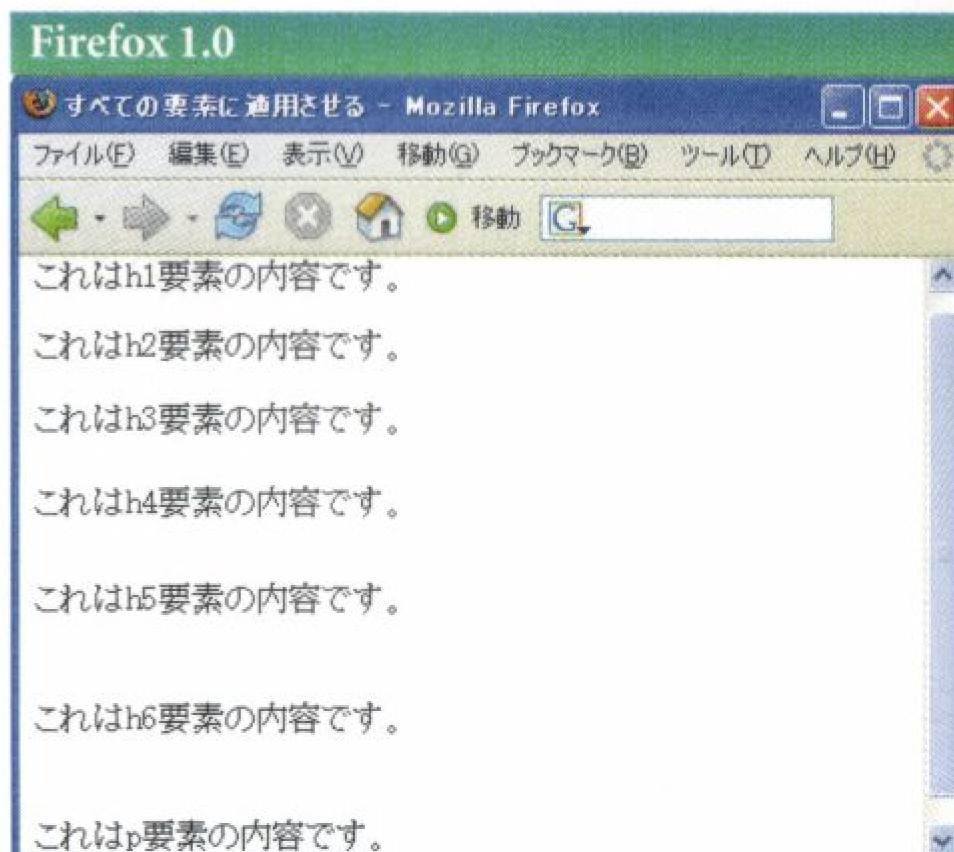
# すべての要素に適用させる

*{ ~ }

Internet Explorer 6.0




Firefox 1.0




すべての要素に対してスタイルを適用させたい場合の書式です。
「*」が単独で使用されている場合は省略できませんが、後に「#ID名」や「.クラス名」などが続いている場合には、「*」を省略することができます。

## Sample

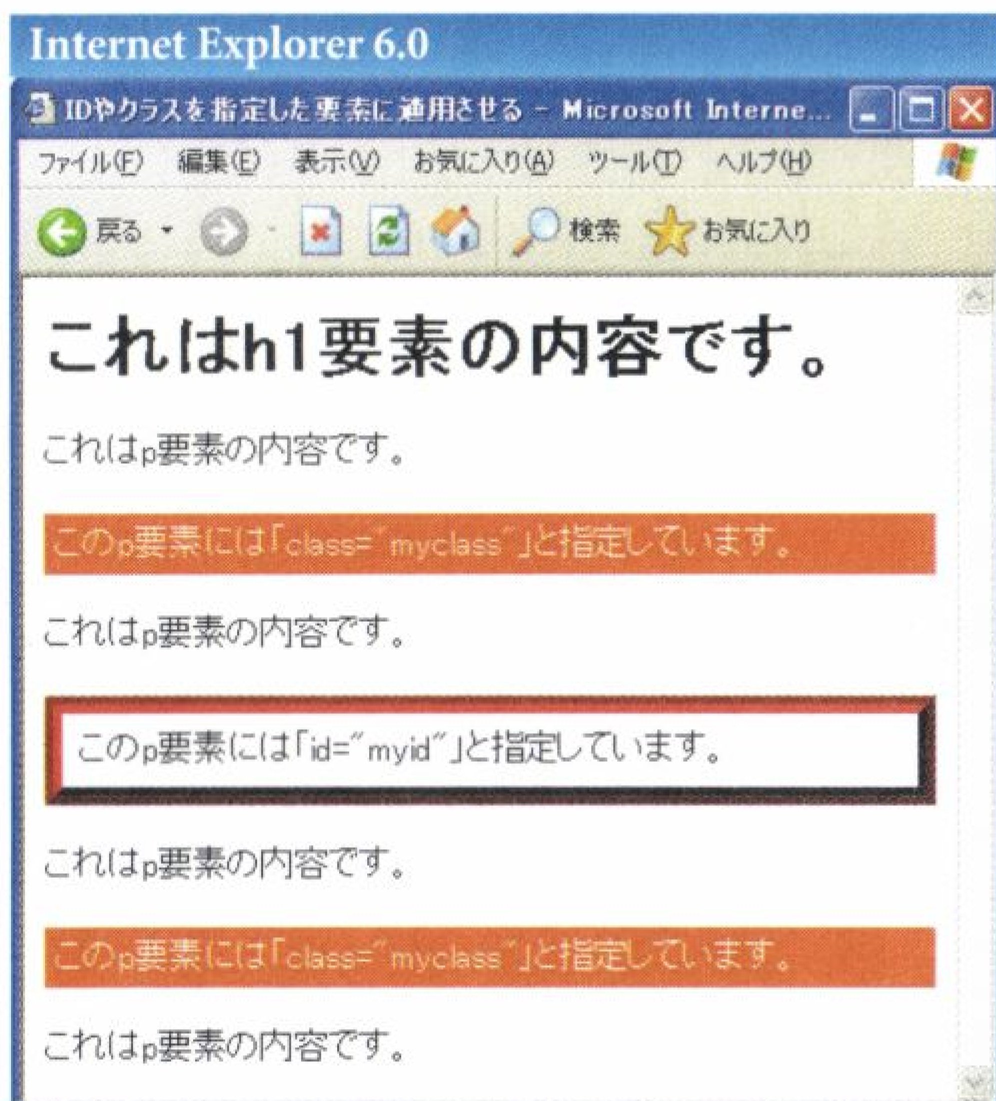
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>すべての要素に適用させる</title>
<style type="text/css">
<!--
* {
  font-size: medium;     /* 文字サイズを標準に */
  font-weight: normal;   /* 文字の太さも標準に */
}
-->
</style>
</head>
<body>
<h1>これはh1要素の内容です。</h1>
<h2>これはh2要素の内容です。</h2>
<h3>これはh3要素の内容です。</h3>
<h4>これはh4要素の内容です。</h4>
<h5>これはh5要素の内容です。</h5>
<h6>これはh6要素の内容です。</h6>
<p>これはp要素の内容です。</p>
</body></html>
```

# IDやクラスを指定した要素に適用させる

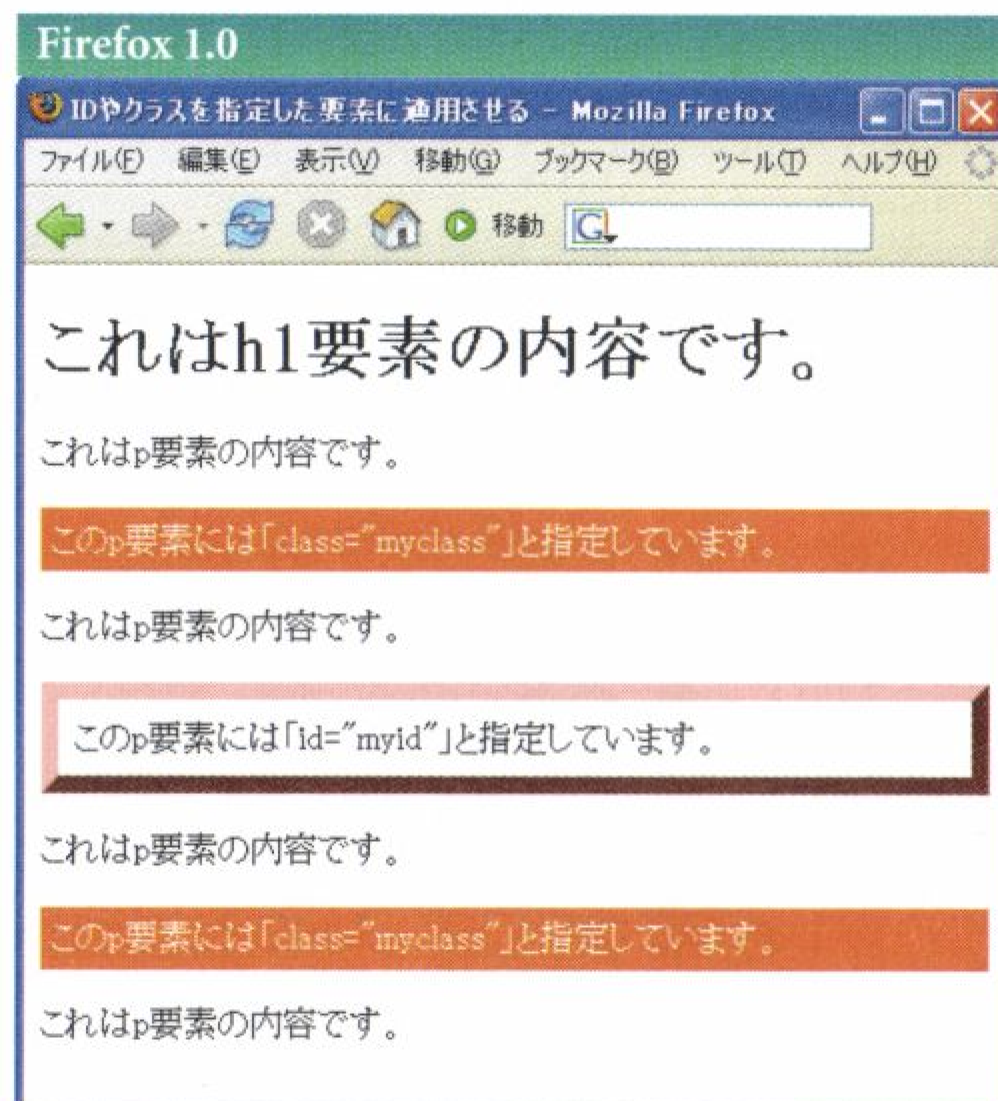
**#ID名 { ~ }**
**.クラス名 { ~ }**
**要素名#ID名 { ~ }**
**要素名.クラス名 { ~ }**



Internet Explorer 6.0




Firefox 1.0




id属性やclass属性で特定の名前が付けられている要素を対象として、スタイルを適用させたい場合の書式です。
「#ID名」と「.クラス名」は、それぞれ「*#ID名」と「*.クラス名」の省略形で、そのID名またはクラス名が指定されているすべての要素が対象となります。「要素名#ID名」または「要素名.クラス名」のように指定すると、そのID名またはクラス名が指定されている「要素名」の要素だけが対象となります。
id属性は、ある要素に対してそのHTML文書内での固有の名前を付けるもので、class属性はその要素の種類名や分類名を表すものです。したがって、ひとつのHTML文書内で同じクラス名は複数箇所に指定できますが、ID名は1ヵ所にしか指定できませんので、注意してください。



## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>IDやクラスを指定した要素に適用させる</title>
<style type="text/css">
```

```
<!--
#myid {
  padding: 0.5em;
  border: outset 8px #ff0000   /* 赤い枠を表示 */
}
p.myclass {
  padding: 0.3em;
  color: #ffffff;              /*文字色を白に */
  background: #ff6600          /* 背景色をオレンジに */
}
-->
</style>
</head>
<body>

<h1 class="myclass">これはh1要素の内容です。</h1>
<p>これはp要素の内容です。</p>
<p class="myclass">
このp要素には「class="myclass"」と指定しています。</p>
<p>これはp要素の内容です。</p>
<p id="myid">
このp要素には「id="myid"」と指定しています。</p>
<p>これはp要素の内容です。</p>
<p class="myclass">
このp要素には「class="myclass"」と指定しています。</p>
<p>これはp要素の内容です。</p>
</body>
</html>
```

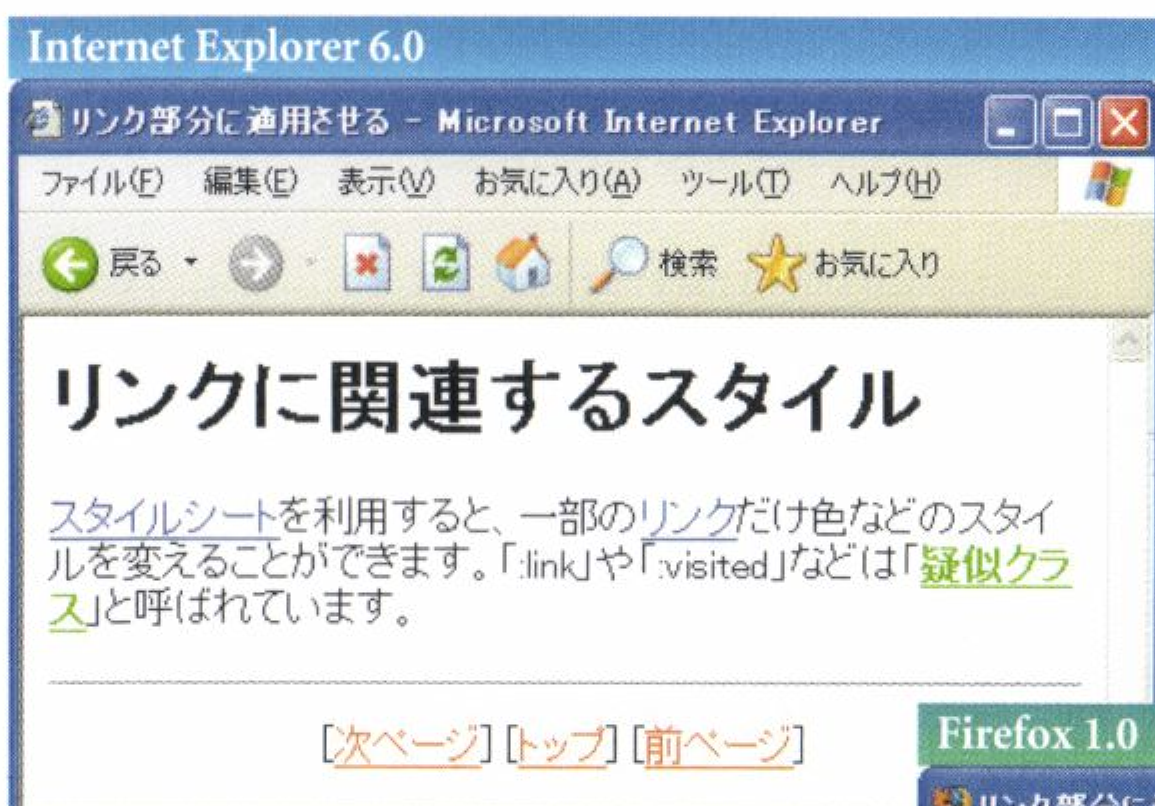
# リンク部分に適用させる

要素名:link { ~ }　←まだ見ていないリンク部分のスタイル
要素名:visited { ~ }　←すでに見たリンク部分のスタイル
要素名:hover { ~ }　←カーソルがその要素の上にある時のスタイル
要素名:active { ~ }　←マウスボタンを押した時のリンク部分のスタイル

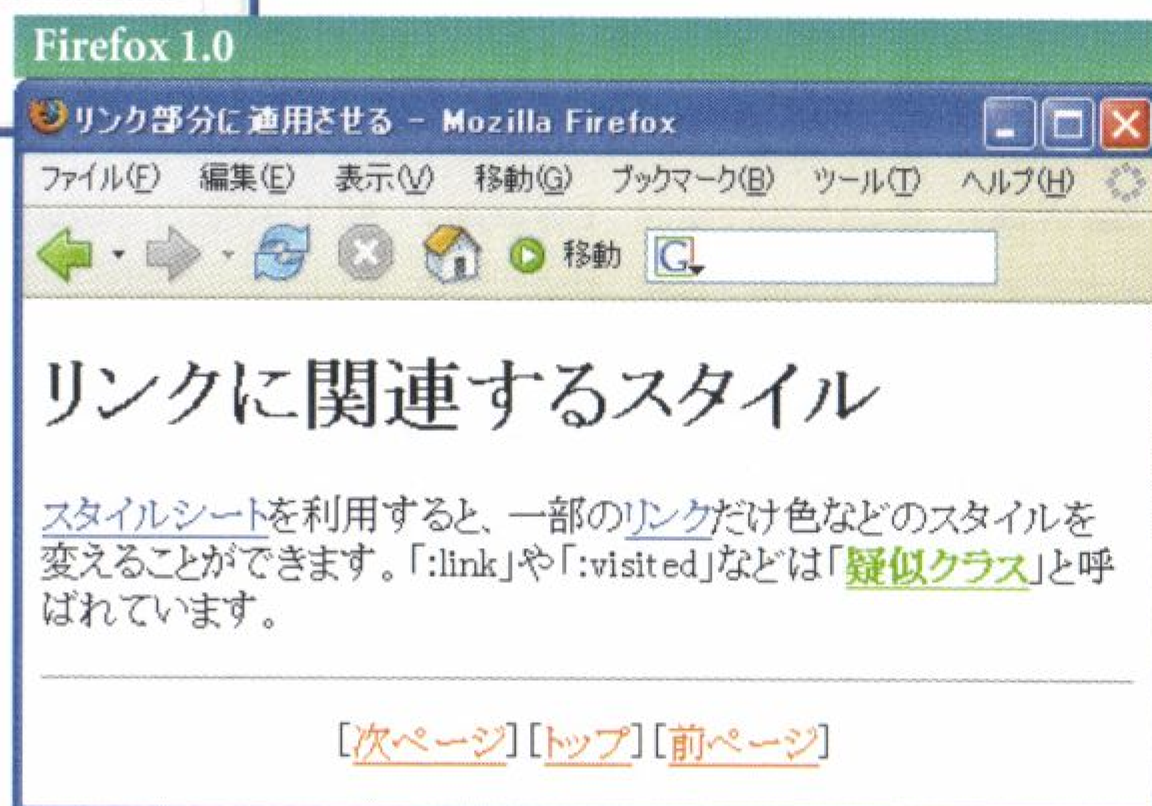
※「要素名」の部分には、「#ID名」や「.クラス名」も指定できます。



Internet Explorer 6.0




Firefox 1.0






リンク部分に対してスタイルを適用させたい場合の書式です。
したがって、通常は「要素名」の部分は「a」になります。「要素名」の部分に「#ID名」や「.クラス名」を指定することで、特定のリンク部分に別のスタイルを適用させることもできます。
これら4種類の状態の指定は、必ず「link」「visited」「hover」「active」の順になるようにしてください。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" ↵
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
<title>リンク部分に適用させる</title>
<style type="text/css">
<!--
/* 普通のリンク色の設定 */
a:link { color: #0000ff; background: #ffffff }
a:visited { color: #000080; background: #ffffff }
a:hover { color: #ff3300; background: #ffffff }
a:active { color: #ff0000; background: #ffffff }

/* 特定のクラスが指定されているリンクだけ色を変える */
a.special { font-weight: bold }
a.special:link { color: #00cc00; background: #ffffff }
a.special:visited { color: #009900; background: #ffffff }
a.special:active { color: #00ff00; background: #ffffff }

.navbar {
  text-align: center;
  border-top: solid 1px #999999;
  padding-top: 1em
}

/* 特定のクラスに含まれているリンクだけ色を変える */
.navbar a:link { color: #ff6600; background: #ffffff }
.navbar a:visited { color: #ff9900; background: #ffffff }
.navbar a:active { color: #ff0000; background: #ffffff }
-->
</style>
</head>
<body>
<h1>リンクに関連するスタイル</h1>
<p>
<a href="css.html">スタイルシート</a>を利用すると、一部の<a href="link.html">リンク</a>だけ色などのスタイルを変えることができます。「:link」や「:visited」などは「<a href="pseudo.html" class="special">疑似クラス</a>」と呼ばれています。
</p>
<p class="navbar">
[<a href="next.html">次ページ</a>]
[<a href="top.html">トップ</a>]
[<a href="prev.html">前ページ</a>]
</p>
</body>
</html>
```


色指定：「CSSについて」の「色の指定方法」(P.180)
要素名：「CSSを適用する対象」の「IDやクラスを指定した要素に適用させる」(P.193)

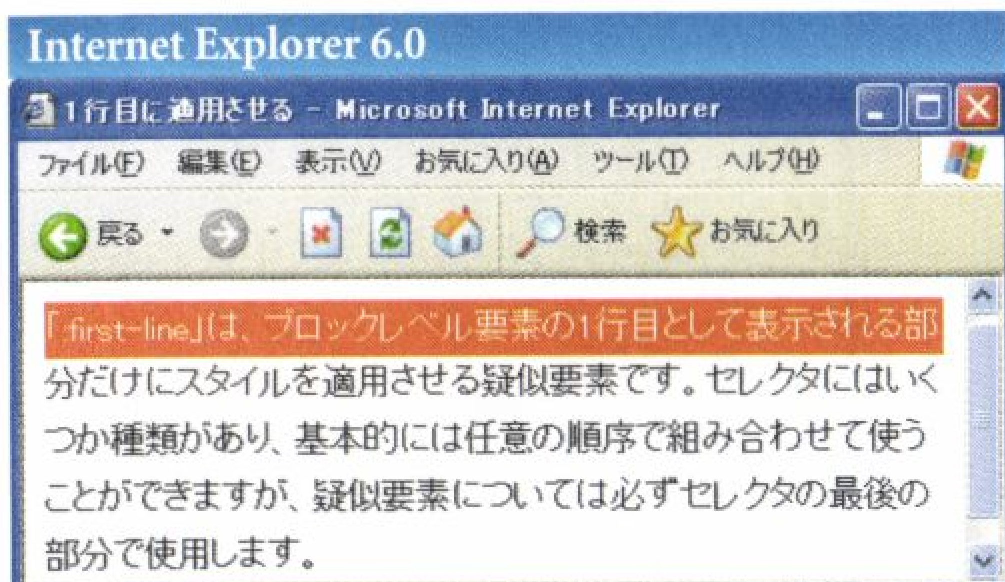
# 1行目に適用させる

**要素名:first-line { ~ }**

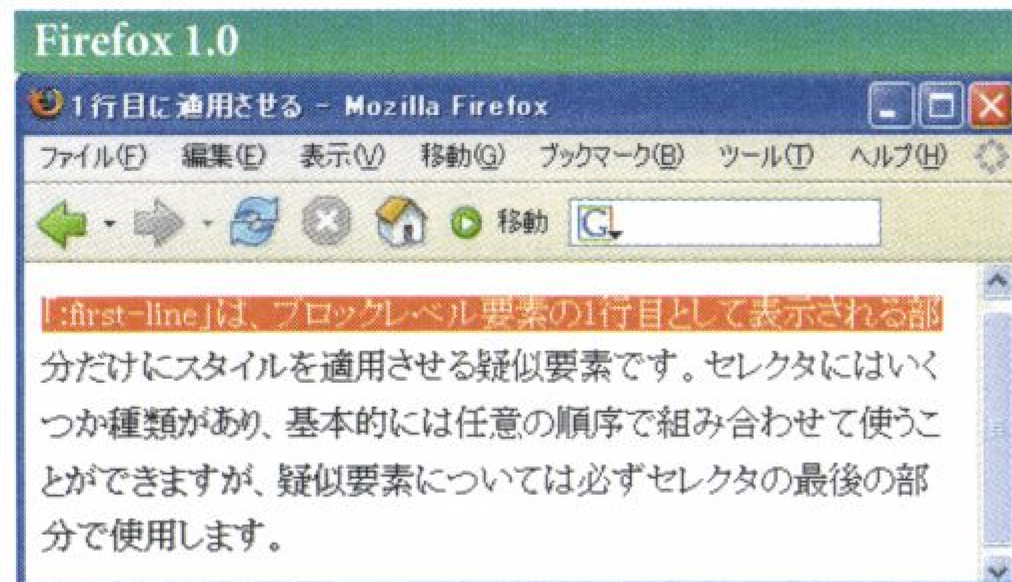
※「要素名」の部分には、「#ID名」や「.クラス名」も指定できます。



Internet Explorer 6.0




Firefox 1.0




指定したブロックレベル要素の1行目として表示される部分だけにスタイルを適用させたい場合の書式です。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>1行目に適用させる</title>
<style type="text/css">
<!--
p { line-height: 1.6 }
p:first-line {
  color: #ffffff;
  background: #ff6600
}
-->
</style>
</head>
<body>
<p>
「:first-line」は、ブロックレベル要素の1行目として表示される部分だけにスタイルを適用させる疑似要素です。セレクタにはいくつか種類があり、基本的には任意の順序で組み合わせて使うことができますが、疑似要素については必ずセレクタの最後の部分で使用します。
</p>
</body>
</html>
```

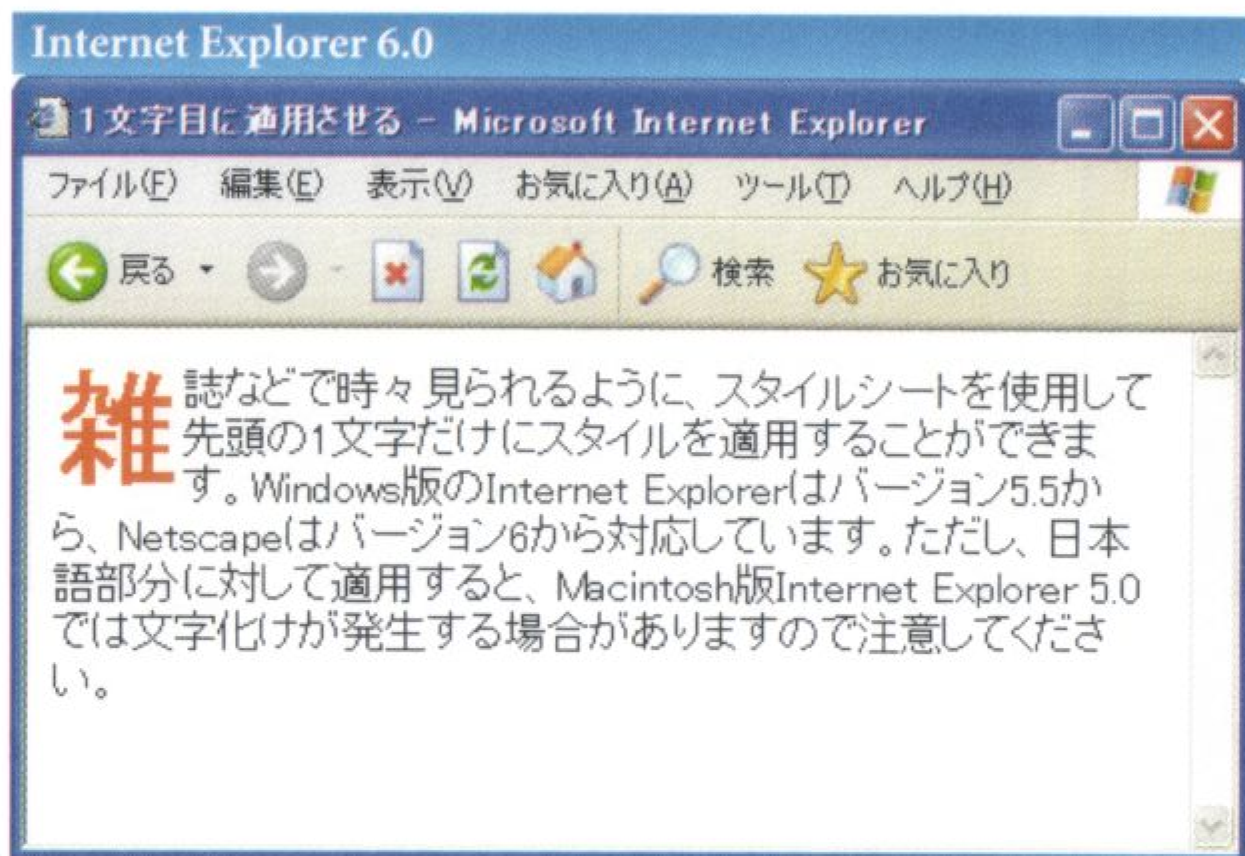

要素名：「CSSを適用する対象」の「IDやクラスを指定した要素に適用させる」(P.193)

# 1文字目に適用させる

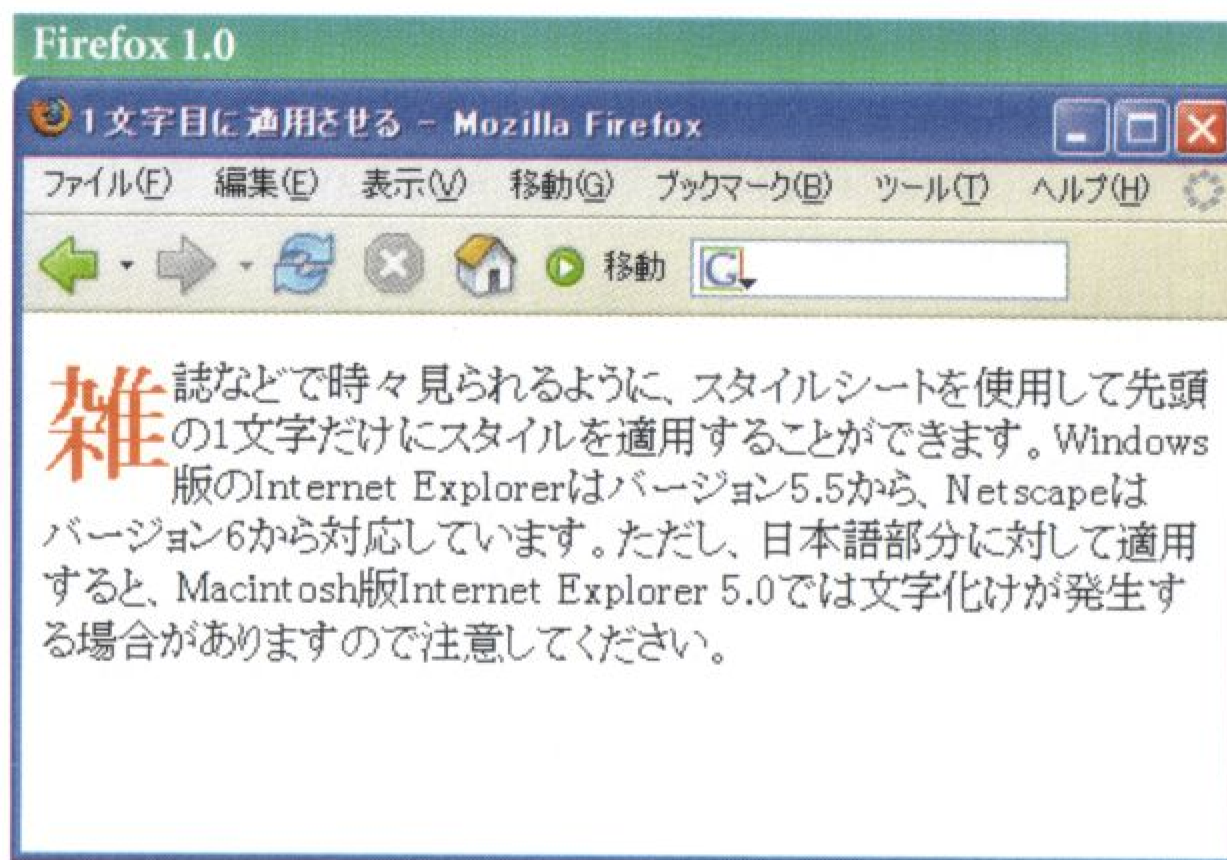
**要素名:first-letter { ~ }**

※「要素名」の部分には、「#ID名」や「.クラス名」も指定できます。

Internet Explorer 6.0




Firefox 1.0




指定したブロックレベル要素の最初の1文字だけにスタイルを適用させたい場合の書式です。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>1文字目に適用させる</title>
<style type="text/css">
<!--
```

```
p:first-letter {
  font-size: 3em;
  float: left;          /* テキストを回り込ませる */
  font-weight: bold;    /* 文字を太くする */
  color: #ff6600;       /* 文字色をオレンジに */
  background: #ffffff
}
-->
</style>
</head>
<body>
<p>
雑誌などで時々見られるように、スタイルシートを使用して先頭の1文字だけにスタイルを適用することができます。Windows版のInternet Explorerはバージョン5.5から、Netscapeはバージョン6から対応しています。ただし、日本語部分に対して適用すると、Macintosh版Internet Explorer 5.0では文字化けが発生する場合がありますので注意してください。
</p>
</body>
</html>
```

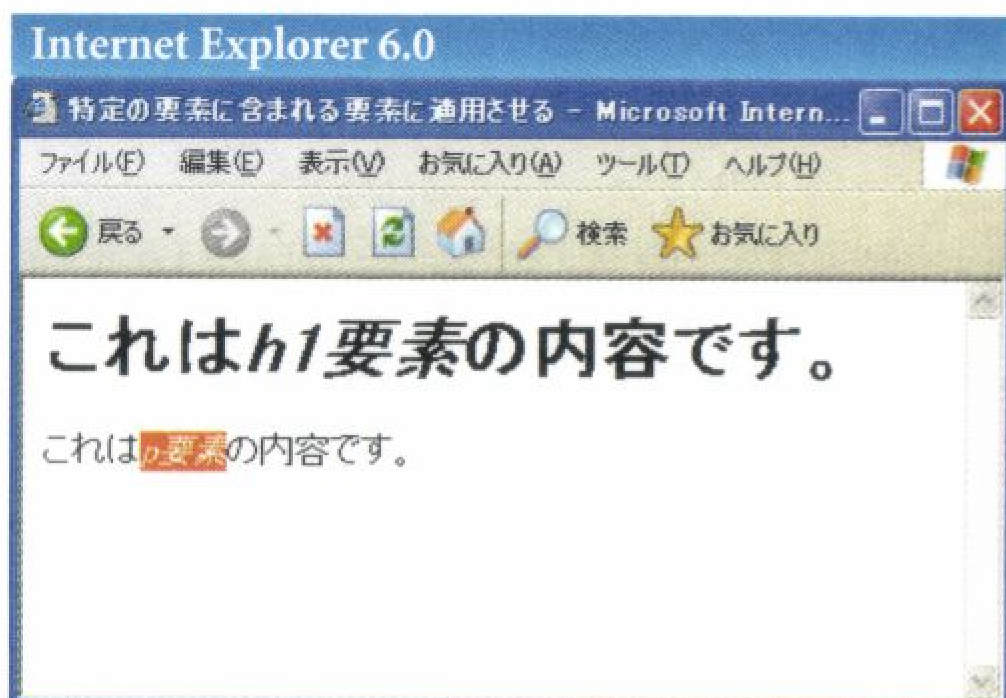

要素名：「CSSを適用する対象」の「IDやクラスを指定した要素に適用させる」（P.193）

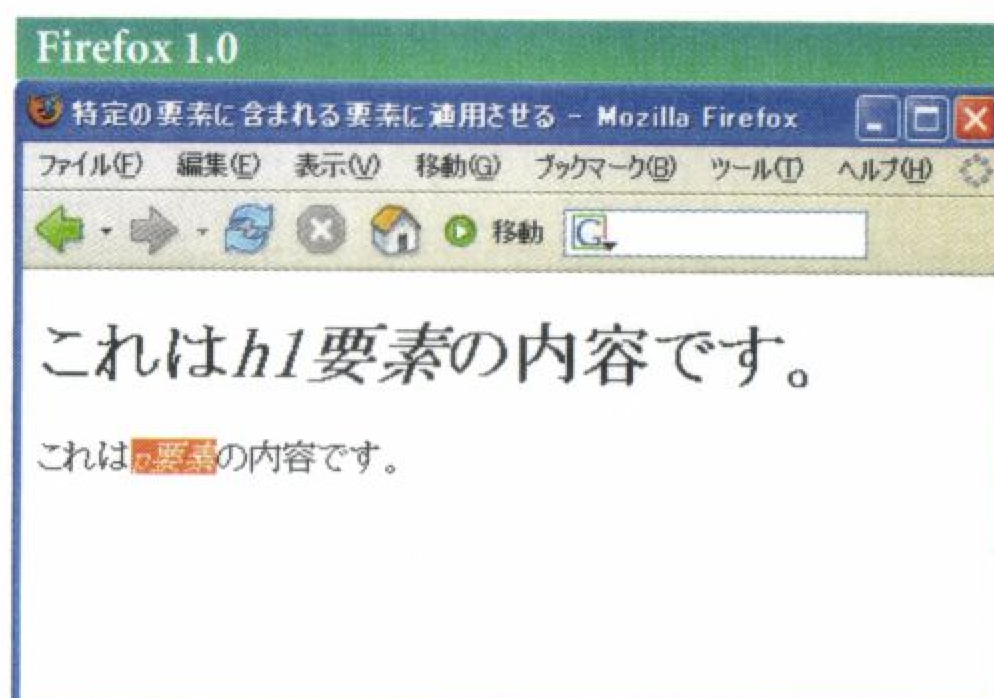
CSS

# 特定の要素に含まれる要素に適用させる

**要素名 要素名 { ~ }**

※「要素名」の部分には、「#ID 名」や「.クラス名」も指定できます。




前の要素の中に含まれる後の要素に対して、スタイルを適用させたい場合の書式です。要素名と要素名の間は、半角スペースで区切ります。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html;
 charset=Shift_JIS">
<title>特定の要素に含まれる要素に適用させる</title>
<style type="text/css">
<!--
p em {
  color: #ffffff;          /* 文字色を白に */
  background: #ff6600      /* 背景色をオレンジに */
}
-->
</style>
</head>
<body>
<h1>これは<em>h1要素</em>の内容です。</h1>
<p>これは<em>p要素</em>の内容です。</p>
</body>
</html>
```


要素名：「CSSを適用する対象」の「IDやクラスを指定した要素に適用させる」(P.193)

# 文字色を指定する

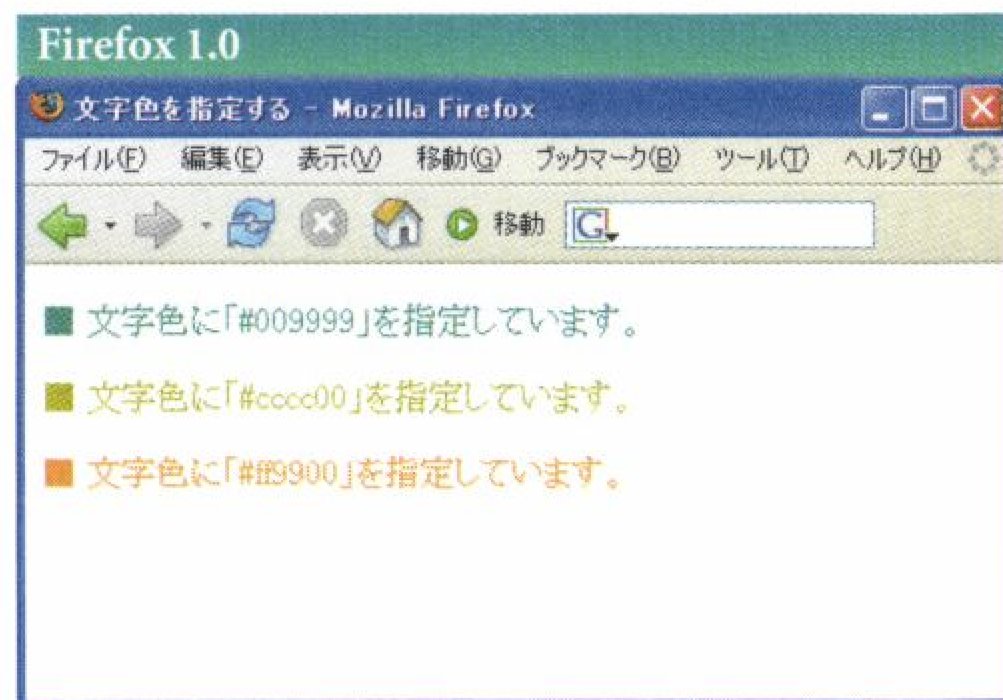
## color: 色指定



Internet Explorer 6.0




Firefox 1.0




color プロパティは、文字の色(前景色)を指定します。
文字色だけを指定していると、ユーザーが設定画面や独自のスタイルシートで背景色を指定している場合などに、文字が読みにくくなってしまう場合があります。そのようなことを避けるために、文字色を指定する場合には背景色も同時に指定するようにしてください。

## Sample

**【CSS】**

```
.type1 {
  color: #009999;
  background: #ffffff
}
.type2 {
  color: #cccc00;
  background: #ffffff
}
.type3 {
  color: #ff9900;
  background: #ffffff
}
```

**【HTML】**

```
<p class="type1">■ 文字色に「#009999」を指定しています。</p>
<p class="type2">■ 文字色に「#cccc00」を指定しています。</p>
<p class="type3">■ 文字色に「#ff9900」を指定しています。</p>
```


色指定：「CSSについて」の「色の指定方法」(P.180)
色指定：巻末付録「カラーチャート1～3」(巻末)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# フォントを指定する

**font-family: フォント名, フォント名, フォント名, …**

フォント名　フォントファミリー名
　　　　　　フォントの種類（serif・sans-serif・cursive・fantasy・monospace）

Internet Explorer 6.0




Firefox 1.0




font-familyプロパティは、フォント名を指定します。
フォント名はひとつでも指定できますが、「,」で区切って複数指定することができます。その場合は、より先（左）に指定されているフォントで、ユーザーの環境で表示可能なものが採用されます。フォント名の中にスペースが含まれている場合は、必ずその前後を「"」または「'」で囲うようにしてください。また、フォント名として、フォントの種類を表すキーワードを指定することもできます。各キーワードは、それぞれ次のような種類を表しています。

| | |
|---|---|
| ・serif | 明朝系（例：Times New Roman, ＭＳ Ｐ明朝） |
| ・sans-serif | ゴシック系（例：Helvetica, ＭＳ Ｐゴシック, Osaka） |
| ・cursive | 草書体・筆記体系（例：Caflisch Script, Adobe Poetica） |
| ・fantasy | 装飾的なフォント（例：Critter, Cottonwood） |
| ・monospace | 等幅フォント（例：ＭＳ ゴシック, Osaka-等幅） |

フォントを指定する

これらは、指定したすべてのフォント名が有効でない場合の最終的な指定として、常に指定しておいたほうがよいでしょう。

## Sample

**【CSS】**

```
h1 { font-family: Impact, sans-serif }
p { font-size: x-large }
.min { font-family: "ＭＳ Ｐ明朝",平成明朝,serif }
.gth { font-family: "ＭＳ Ｐゴシック",Osaka,sans-serif }
.com { font-family: "Comic Sans MS",sans-serif }
```

**【HTML】**

```
<h1>font-family</h1>
<p class="min">この部分は明朝体に設定しています。</p>
<p class="gth">この部分はゴシック体に設定しています。</p>
<p class="com">Comic Sans MS</p>
```

font：「フォント」の「フォント関係をまとめて指定する」(P.210)
フォント名：巻末付録「フォント表示見本」(巻末)

### コラム

#### フォントファミリーとは？

Windowsを使用している方であれば、「Times New Roman」というフォントの名前を目にしたことがあると思います。このフォントには、元となるひとつのフォントしかないわけではなく、太字用にデザインされた「Times New Roman Bold」、イタリック用にデザインされた「Times New Roman Italic」、太字のイタリックとしてデザインされた「Times New Roman Bold Italic」という別々のフォントが用意されています。フォントは計算によって太くしたり斜めにしたりすることもできますが、あらかじめ専用にデザインしたフォントを用意しておくことで、より美しい表示や印刷が可能になるわけです。

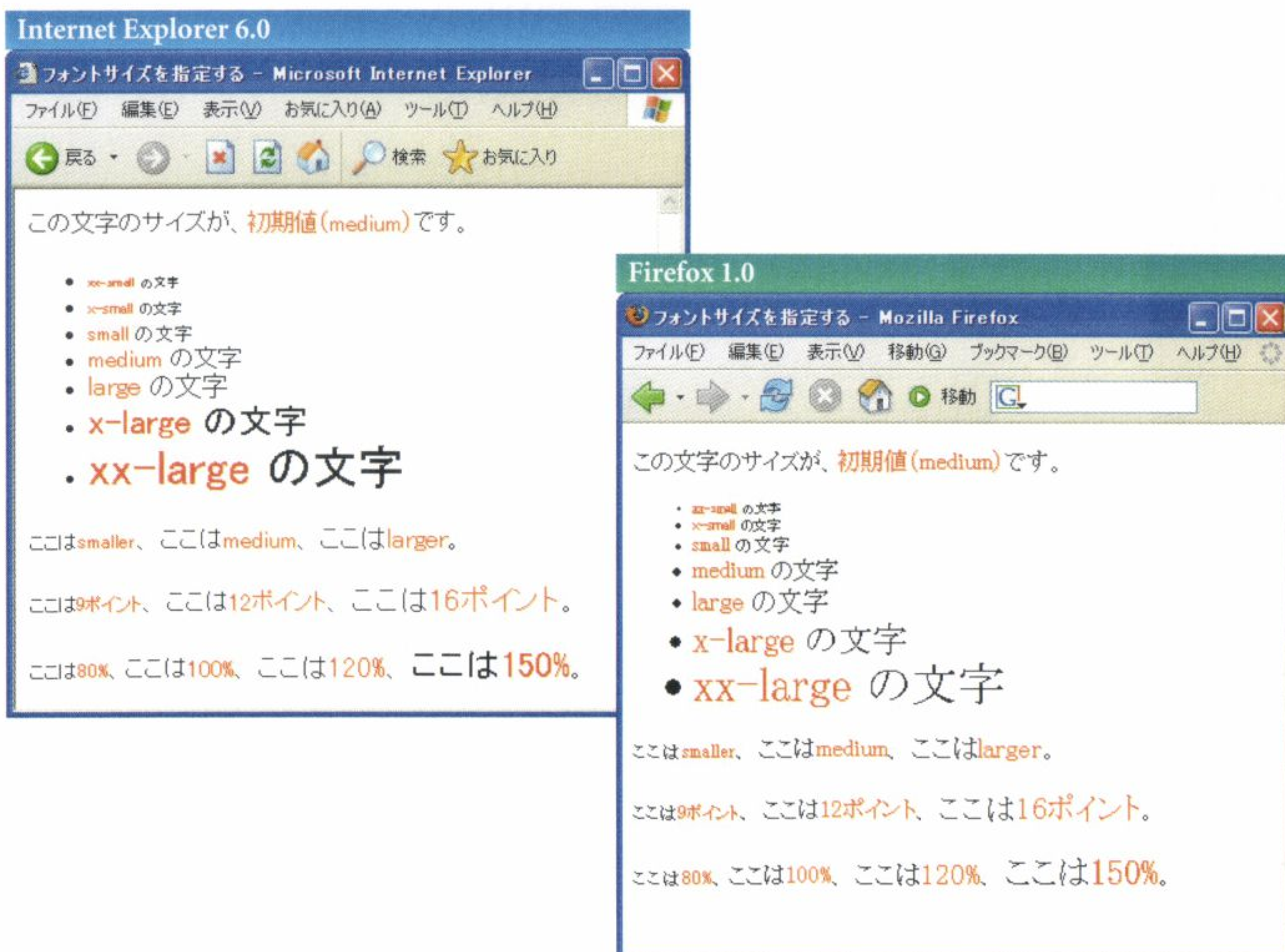
このように、同じ種類でありながら別々のスタイル専用にデザインされたフォントを、まとめて「フォントファミリー」と呼んでいます。上記の例では「Times New Roman」がフォントファミリー名ということになります。

CSS

# フォントサイズを指定する

**font-size: サイズ**

| | |
|---|---|
| サイズ | 単位付きの数値・%・smaller・larger<br>xx-small・x-small・small・medium・large・x-large・xx-large |




font-size プロパティは、フォントのサイズを指定します。
%で指定した場合は、指定した要素の親である要素のフォントサイズに対する割合になります。また、キーワードを利用することによって、「xx-small」から「xx-large」までの7段階のサイズを表現することができます。この場合、「medium」が標準のサイズです。「smaller」と「larger」は、親要素のフォントサイズに対して、それぞれ1段階小さいサイズと1段階大きいサイズに設定します。なお、Windows版のInternet Explorer 5.5までとInternet Explorer 6.0の互換モード、Macintosh版のInternet Explorer 4.Xと Netscape Navigator 4.Xでは、「small」が標準のサイズになっていますので、注意してください。

## Sample

**【CSS】**

```
.xxs { font-size: xx-small }
.xs  { font-size: x-small }
.s   { font-size: small }
```

```
.m   { font-size: medium }
.l   { font-size: large }
.xl  { font-size: x-large }
.xxl { font-size: xx-large }
.smaller { font-size: smaller }
.larger  { font-size: larger }
.s09 { font-size: 9pt }
.s12 { font-size: 12pt }
.s16 { font-size: 16pt }
.p080 { font-size: 80% }
.p120 { font-size: 120% }
.p150 { font-size: 150% }
em {
  color: #ff3300;
  background-color: #ffffff;
  font-style: normal
}
```

**【HTML】**

```
<p>この文字のサイズが、<em>初期値(medium)</em>です。</p>
<ul>
<li class="xxs"><em>xx-small</em> の文字</li>
<li class="xs"><em>x-small</em> の文字</li>
<li class="s"><em>small</em> の文字</li>
<li class="m"><em>medium</em> の文字</li>
<li class="l"><em>large</em> の文字</li>
<li class="xl"><em>x-large</em> の文字</li>
<li class="xxl"><em>xx-large</em> の文字</li>
</ul>
<p>
<span class="smaller">ここは<em>smaller</em></span>、
<span class="m">ここは<em>medium</em></span>、
<span class="larger">ここは<em>larger</em></span>。
</p>
<p>
<span class="s09">ここは<em>9ポイント</em></span>、
<span class="s12">ここは<em>12ポイント</em></span>、
<span class="s16">ここは<em>16ポイント</em></span>。
</p>
<p>
<span class="p080">ここは<em>80%</em></span>、
ここは<em>100%</em>、
<span class="p120">ここは<em>120%</em></span>、
<span class="p150">ここは<em>150%</em></span>。
</p>
```

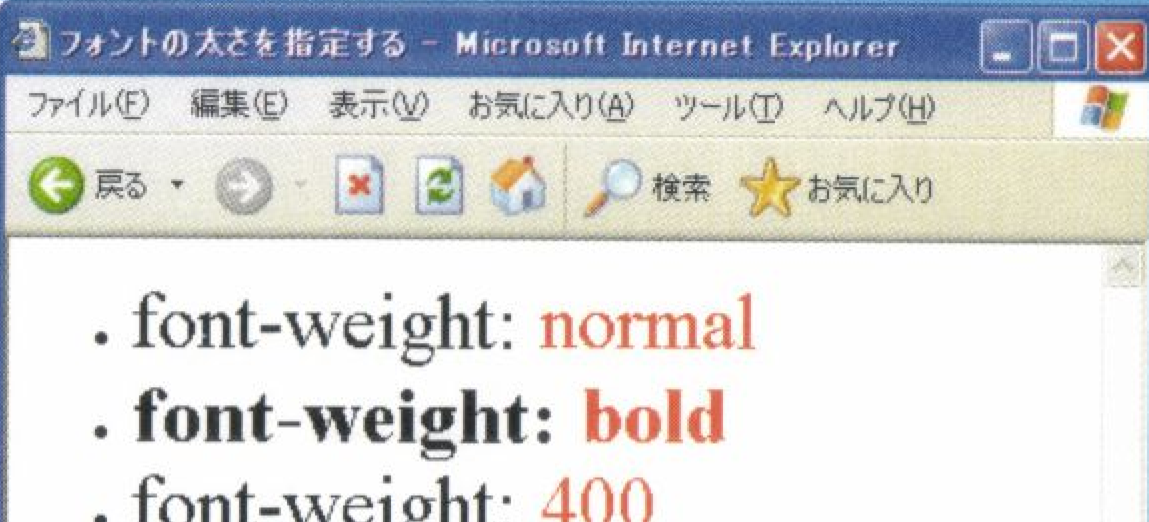

HTML：TIPS「新しいブラウザは<!DOCTYPE>の書き方で表示が変わる！」(P.9)
コラム「フォントサイズは相対的な単位で指定する」(P.207)
font：「フォント」の「フォント関係をまとめて指定する」(P.210)

# フォントの太さを指定する

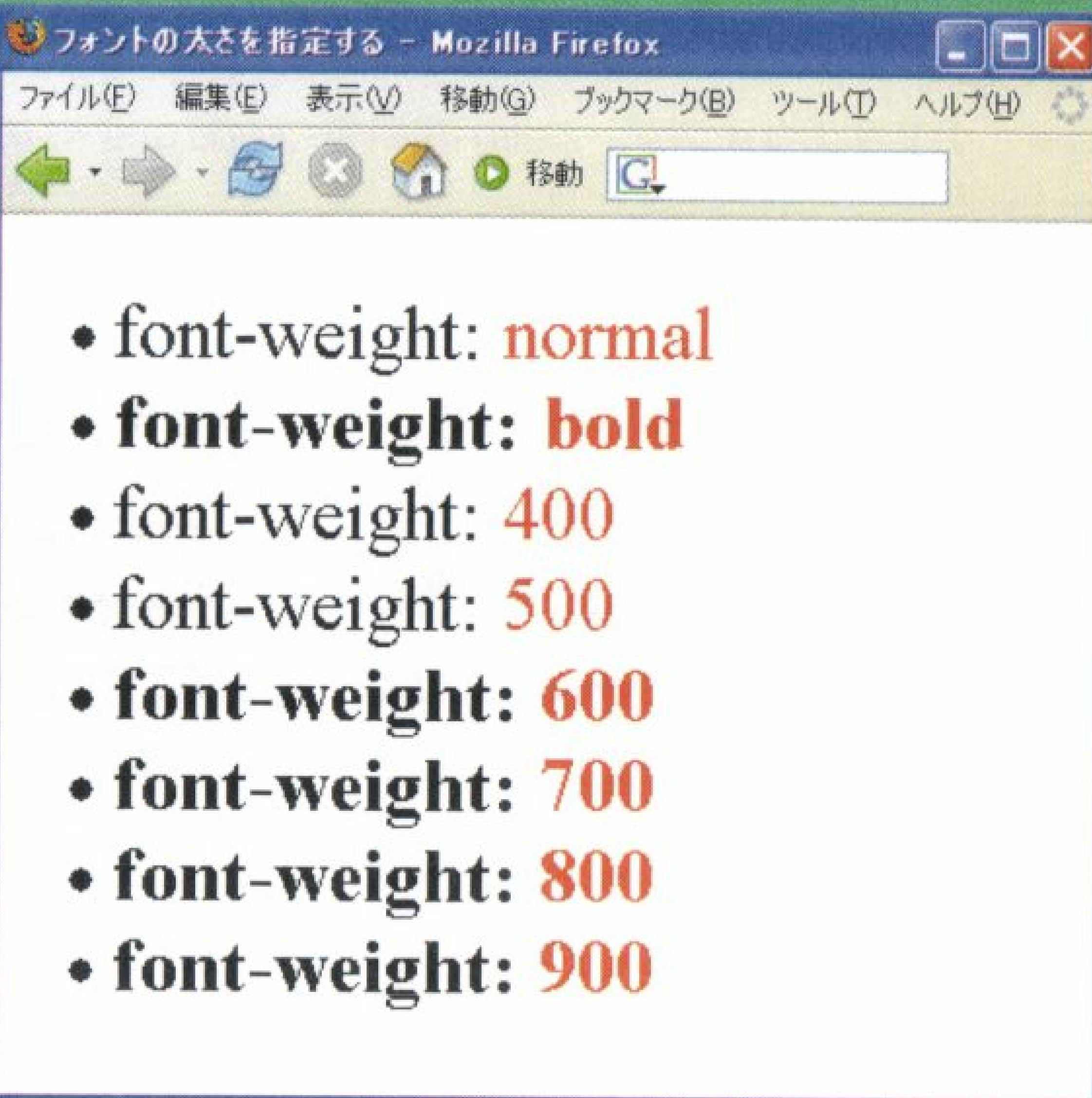
## font-weight: 太さ

| | |
|---|---|
| 太さ | normal・bold・lighter・bolder<br>100・200・300・400・500・600・700・800・900 |

Internet Explorer 6.0




Firefox 1.0




font-weightプロパティは、フォントの太さを指定します。
「bold」を指定すると一般的な太字になります。同じフォントファミリー中に異なる太さのフォントがある場合、「lighter」は1段階細いフォントに、「bolder」は1段階太いフォントに設定されます。また、太さは100～900の数値で指定することもできます。この場合、標準の太さ「normal」は400で、「bold」は700に相当します。

### Sample

**【CSS】**

```
ul {
  font-size: xx-large;
  font-family: "Times New Roman", Times, serif
}
```

```
#bold { font-weight: bold }
#w400 { font-weight: 400 }
#w500 { font-weight: 500 }
#w600 { font-weight: 600 }
#w700 { font-weight: 700 }
#w800 { font-weight: 800 }
#w900 { font-weight: 900 }
em {
  color: #ff3300;
  background-color: #ffffff;
  font-style: normal
}
```

【HTML】

```
<ul>
<li>font-weight: <em>normal</em></li>
<li id="bold">font-weight: <em>bold</em></li>
<li id="w400">font-weight: <em>400</em></li>
<li id="w500">font-weight: <em>500</em></li>
<li id="w600">font-weight: <em>600</em></li>
<li id="w700">font-weight: <em>700</em></li>
<li id="w800">font-weight: <em>800</em></li>
<li id="w900">font-weight: <em>900</em></li>
</ul>
```

font：「フォント」の「フォント関係をまとめて指定する」(P.210)

コラム

### フォントサイズは相対的な単位で指定する

ブラウザの種類やその設定にもよりますが、フォントサイズの指定に「pt」や「px」などの単位を使うと、ユーザー側で文字の大きさを変更することができなくなる場合があります。これは逆に言えば、フォントサイズを固定したつもりでいても、環境によっては大きさが変更される場合もあるということです。

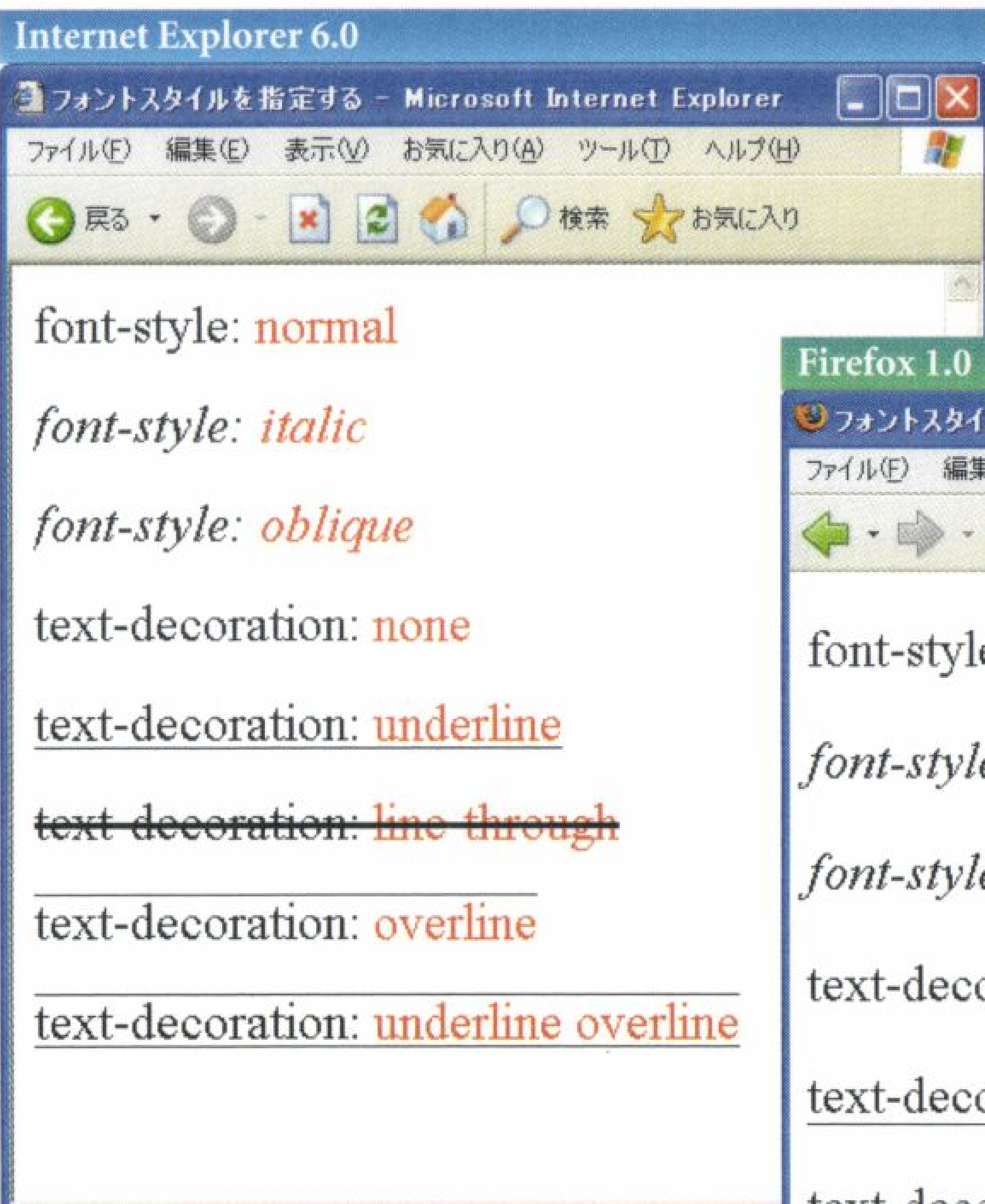
したがって、フォントサイズの指定に「pt」や「px」などの単位を使っても必ずしもその大きさで表示されるとは限らないし、もし表示サイズが固定されたとしても、それはユーザーから「必要に応じて文字の大きさを変更する」機能を奪ってしまうことになります。フォントサイズを指定する場合には、できるだけ「em（その部分の本来のフォントサイズを1とする単位）」や「%」などの相対的な単位を使って、必要に応じて文字の大きさを変更できるようにしておきましょう。

# フォントスタイルを指定する

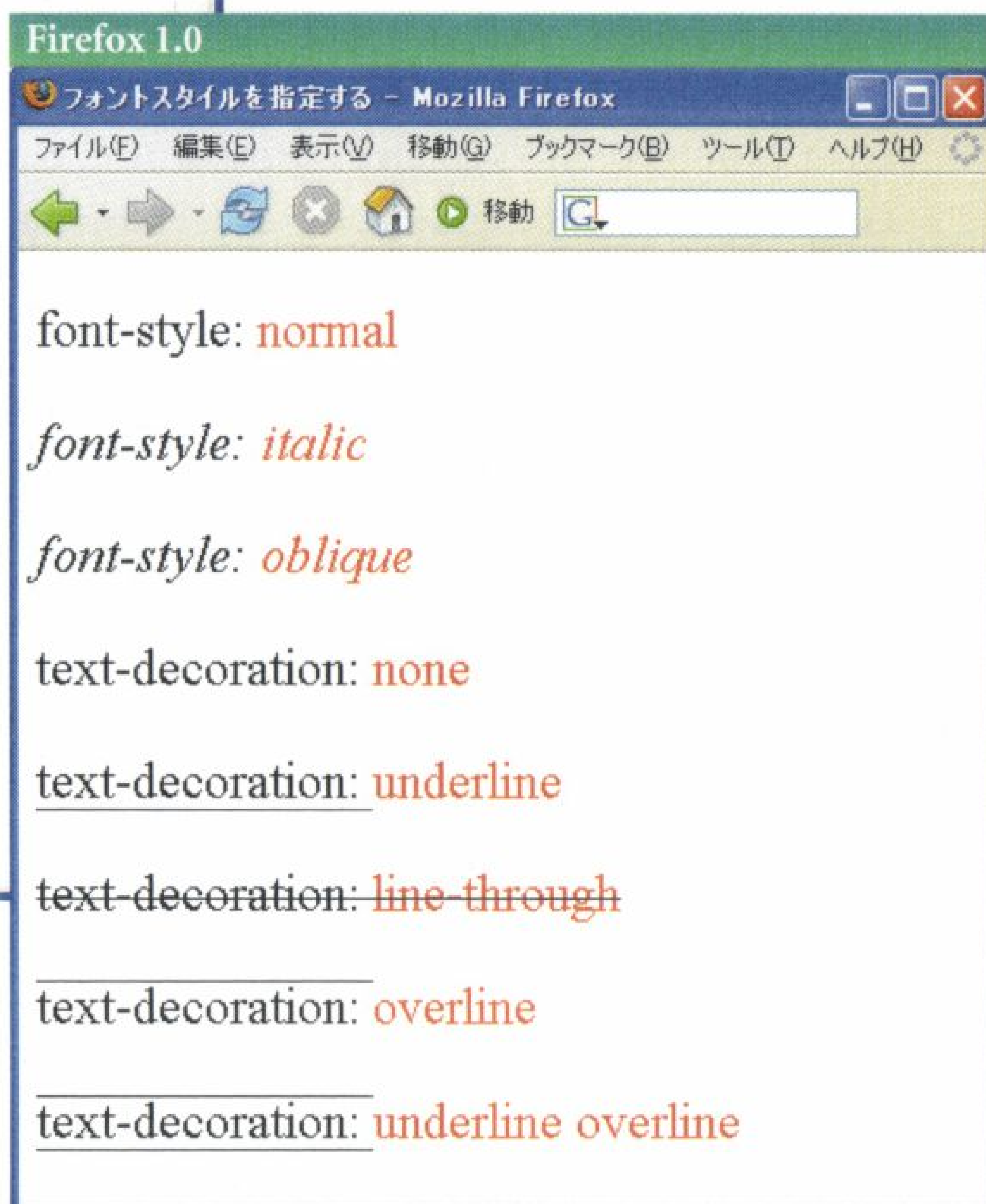
**font-style: *斜体***
**text-decoration: *装飾***

| | |
|---|---|
| 斜体 | normal・italic・oblique |
| 装飾 | none・underline・overline・line-through・blink |

Internet Explorer 6.0




Firefox 1.0




font-style プロパティはフォントを斜体にするかどうかを、text-decoration プロパティはフォントの装飾を指定します。
ここでいう「italic」とは、通常の文字よりも続け書きに近い草書体風にデザインされた斜体のフォントを指します。「oblique」とは、標準のフォントを単純に斜めにしたものです。一般的な環境では、「italic」も「oblique」も同じように斜体で表示されます。
「underline」「overline」「line-through」は、それぞれ文字に対して「下線」「上線」「取消線」を付け、「blink」は文字を点滅させます。これらの値は、半角スペースで区切って複数同時に指定することもできます。
斜体を標準に戻すためにはfont-style プロパティに「normal」を、装飾をなくするためにはtext-decoration プロパティに「none」を指定します。

## Sample

**【CSS】**

```
p {
  font-size: x-large;
  font-family: "Times New Roman", Times, serif
}
#itlc { font-style: italic }
#oblq { font-style: oblique }
#udln { text-decoration: underline }
#lnth { text-decoration: line-through }
#ovln { text-decoration: overline }
#udov { text-decoration: underline overline }
.keywd {
  color: #ff3300;
  background-color: #ffffff
}
```

**【HTML】**

```
<p>
font-style: <span class="keywd">normal</span>
</p>
<p id="itlc">
font-style: <span class="keywd">italic</span>
</p>
<p id="oblq">
font-style: <span class="keywd">oblique</span>
</p>
<p>
text-decoration: <span class="keywd">none</span>
</p>
<p id="udln">
text-decoration: <span class="keywd">underline</span>
</p>
<p id="lnth">
text-decoration: <span class="keywd">line-through</span>
</p>
<p id="ovln">
text-decoration: <span class="keywd">overline</span>
</p>
<p id="udov">
text-decoration: <span class="keywd">underline overline</span>
</p>
```


font：「フォント」の「フォント関係をまとめて指定する」(P.210)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# フォント関係をまとめて指定する

**font: 斜体 太さ サイズ/行間 フォント名**

| | |
|---|---|
| 斜体 | font-style (P.208) で指定できる値 |
| 太さ | font-weight (P.206) で指定できる値 |
| サイズ | font-size (P.204) で指定できる値 (省略不可) |
| 行間 | line-height (P.212) で指定できる値 (値の前に「/」が必要) |
| フォント名 | font-family (P.202) で指定できる値 (省略不可) |

Internet Explorer 6.0




Firefox 1.0




font プロパティで、フォント関連のプロパティの値をまとめて指定します。
サイズと行間の間は「/」で区切りますが、それ以外の値は半角スペースで区切ってください。また、値は基本的に上に示した順序で指定しますが、サイズとフォント名以外は省略することができます。

## Sample

**【CSS】**

```
body {
  margin: 0;
  color: #ffffff;
  background: #009900 url(back.jpg)
}
h1, h2 {
  text-align: center;
  margin: 0
}
h1 {
  font: italic bold 6em "Times New Roman",Times,serif
}
h2 {
  font: 1.5em Arial,sans-serif;
  color: #ff9900;
  background: transparent
}
p {
  font: 12pt/200% "ＭＳ Ｐ明朝",平成明朝,serif;
  margin: 1em 2em;
  padding: 1em;
  color: #ffffff;
  background: #000000
}
```

**【HTML】**

```
<h1>-font-</h1>
<h2>[ shorthand font property ]</h2>
<p>
このプロパティを利用すると、フォントに関連する複数のプロパティを一度に設定することができ
ます。
</p>
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

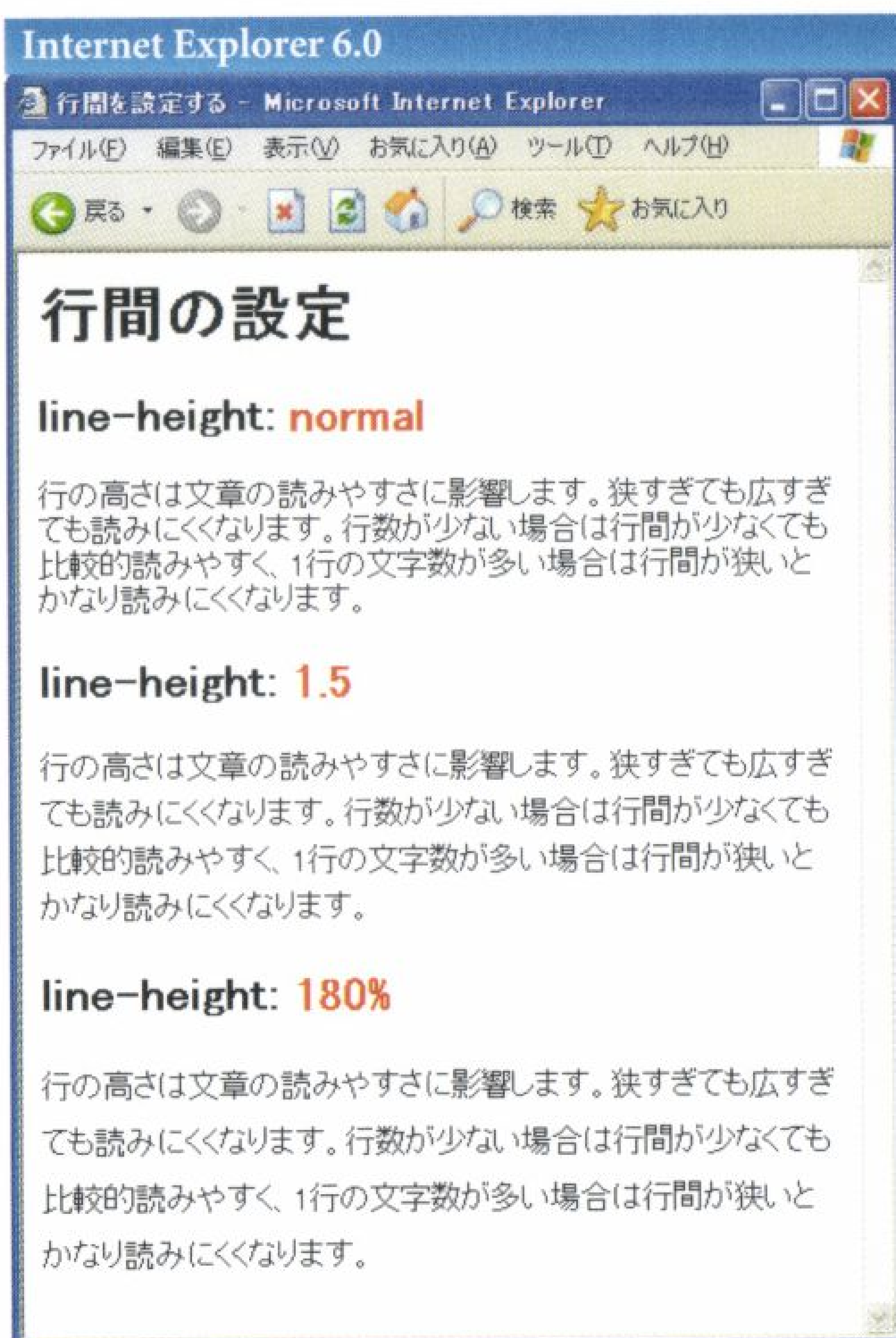
フォントを指定する

# 行間を設定する

## line-height: 行の高さ

| | |
|---|---|
| 行の高さ | normal・数値・単位付きの数値・% |

Internet Explorer 6.0




Firefox 1.0




line-height プロパティは、行の高さを設定します。
単位を付けないで数値だけを指定した場合は、フォントサイズにその値を掛けた高さに設定されます。%による指定も、フォントサイズに対する割合になります。マイナスの値は指定できないことになっていますので、注意してください。

### Sample

【CSS】

```
#sample1 { line-height: normal }
#sample2 { line-height: 1.5 }
#sample3 { line-height: 180% }
em {
  color: #ff3300;
  background-color: #ffffff;
  font-style: normal
}
```

【HTML】
```
<h1>行間の設定</h1>

<h2>line-height: <em>normal</em></h2>
<p id="sample1">
行の高さは文章の読みやすさに影響します。狭すぎても広すぎても読みにくくなります。行数が少ない場合は行間が少なくても比較的読みやすく、1行の文字数が多い場合は行間が狭いとかなり読みにくくなります。
</p>
<h2>line-height: <em>1.5</em></h2>
<p id="sample2">
行の高さは文章の読みやすさに影響します。狭すぎても広すぎても読みにくくなります。行数が少ない場合は行間が少なくても比較的読みやすく、1行の文字数が多い場合は行間が狭いとかなり読みにくくなります。
</p>
<h2>line-height: <em>180%</em></h2>
<p id="sample3">
行の高さは文章の読みやすさに影響します。狭すぎても広すぎても読みにくくなります。行数が少ない場合は行間が少なくても比較的読みやすく、1行の文字数が多い場合は行間が狭いとかなり読みにくくなります。
</p>
```


font：「フォント」の「フォント関係をまとめて指定する」(P.210)

## コラム

### line-heightプロパティのバグに注意！

Internet Explorer 3.XやNetscape Navigator 4.Xには、line-heightプロパティに関するバグがたくさんあります。たとえば、Internet Explorer 3.Xの場合は単位を付けない数値を指定するとピクセル数として認識されますし、Netscape Navigator 4.Xでは絶対的な単位を指定していると印刷で問題が発生する場合もあります。

中でも特に注意する必要があるのが、Netscape Navigator 4.Xではline-heightを指定した範囲に画像が含まれていると、テキストの上に画像が重なって表示されるというバグです。Netscape Navigator 4.Xのユーザーから「文字の上に画像が重なって読めない」という苦情が来たら、まずはline-heightを疑ってみましょう。

CSS

# 行揃えを指定する

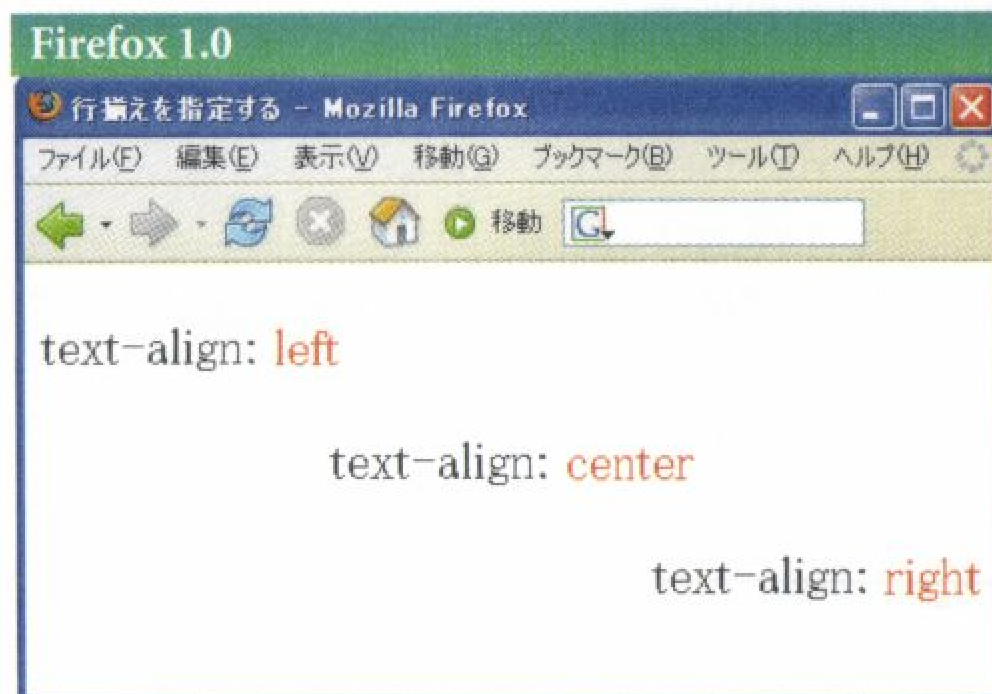
**text-align:** *行揃え位置*

行揃え位置　left・right・center

Internet Explorer 6.0




Firefox 1.0




text-align プロパティは、行揃えを設定します。
「left」「right」「center」は、それぞれ「左揃え」「右揃え」「中央揃え」を表しています。このプロパティは、ブロックレベル要素に対して指定して、その内容の行揃えを設定するものです。ブロックレベル要素のボックス自体をセンタリングしたい場合には、左右のマージンを「auto」に設定してください。

## Sample

**【CSS】**

```
p { font-size: x-large }
#sample1 { text-align: left }
#sample2 { text-align: center }
#sample3 { text-align: right }
em {
  font-style: normal;
  color: #ff3300;
  background-color: #ffffff
}
```

**【HTML】**

```
<p id="sample1">text-align: <em>left</em></p>
<p id="sample2">text-align: <em>center</em></p>
<p id="sample3">text-align: <em>right</em></p>
```

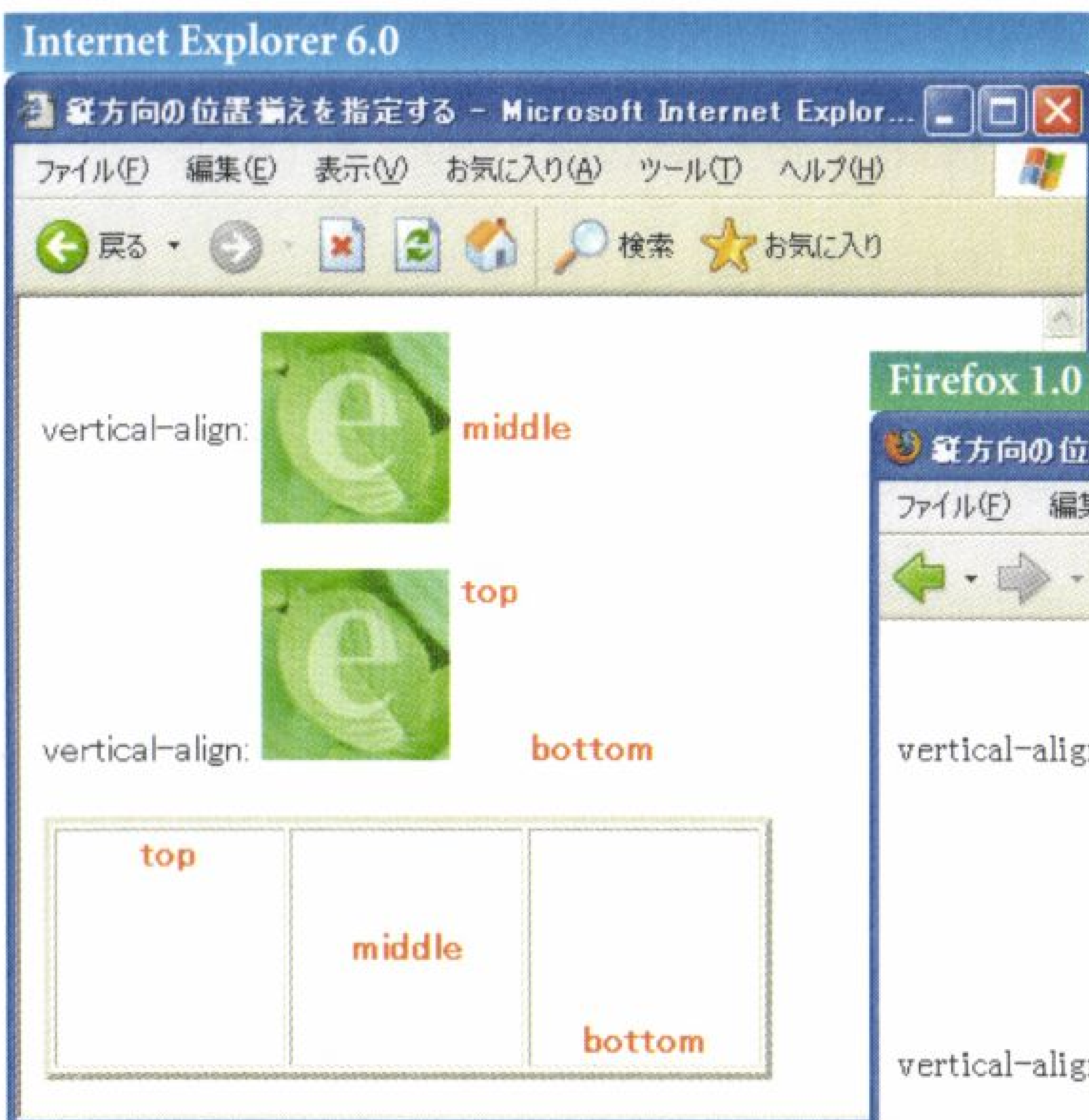
margin：「表示と配置」の「センタリングする」(P.256)

# 縦方向の位置を指定する

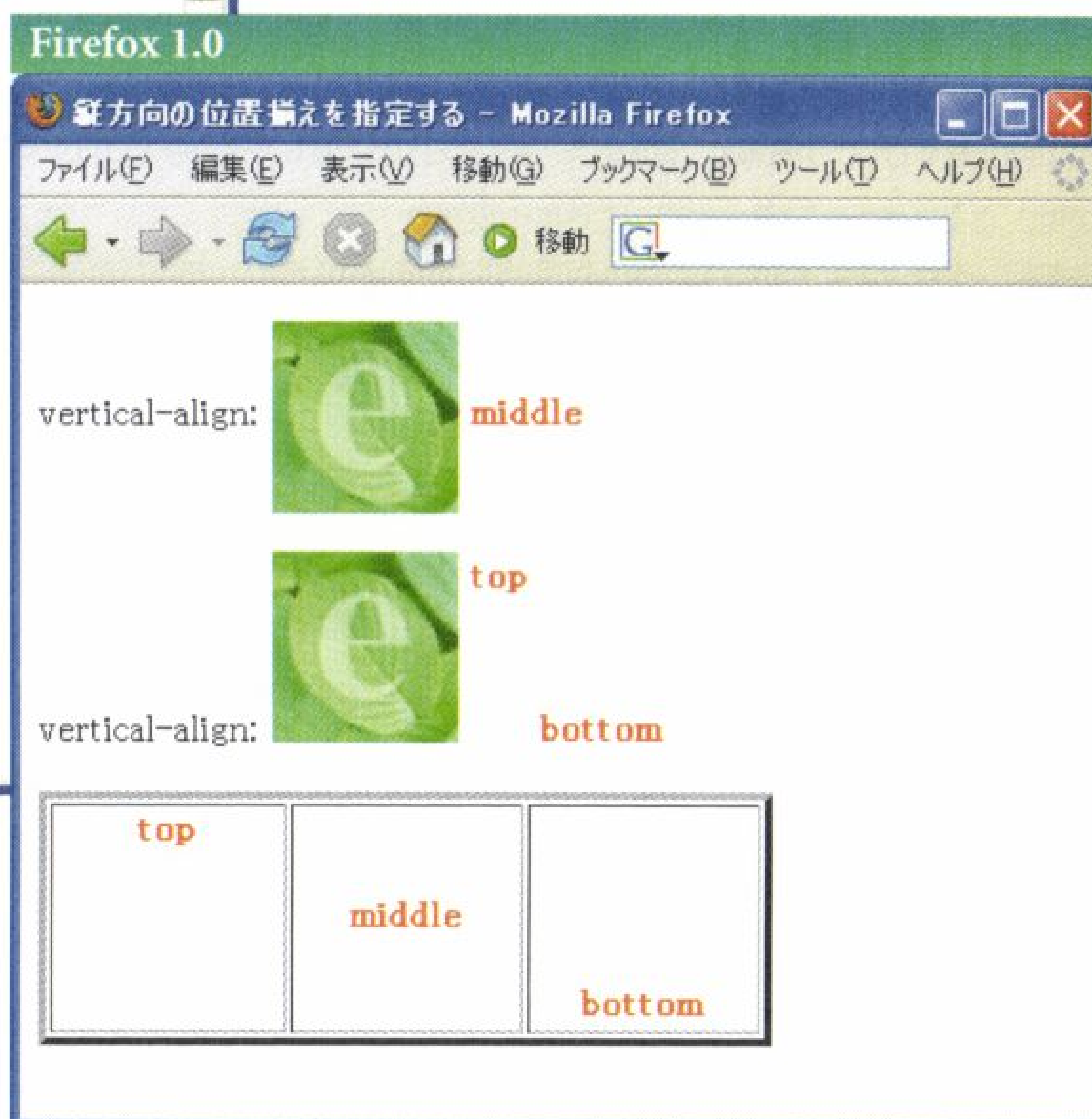
## vertical-align: 縦方向の位置

| 【縦方向の位置】 | |
|---|---|
| top | 上をその行の上に合わせる |
| middle | 中心をその行の「ベースライン + 小文字xの半分の高さ」に合わせる |
| bottom | 下をその行の下に合わせる |
| baseline | ベースラインをその行のベースラインに合わせる（初期値） |
| text-top | 上をその行のフォントの上に合わせる |
| text-bottom | 下をその行のフォントの下に合わせる |
| super | ベースラインをその行の上付き文字の位置に合わせる |
| sub | ベースラインをその行の下付き文字の位置に合わせる |
| 単位付きの数値 | その行のベースラインを0とした値の+- |
| % | その行のベースラインを0とした行の高さに対する割合 |

Internet Explorer 6.0




Firefox 1.0




vertical-alignプロパティは、指定されたインライン要素が表示される行の中での、縦方向の位置を指定します。

このプロパティは行全体の縦の位置揃えを指定するものではなく、指定されたインライン要素を、その行の中で縦方向のどの位置に配置するかを指定するものです。

なお、このプロパティをth要素またはtd要素に指定した場合には、「top」「middle」「bottom」は、それぞれセル内の「上」「中心」「下」に揃えられ、「baseline」を指定した場合にはセル内の最初の行のベースラインが揃えられます。

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

テキストを設定する

## Sample

**【CSS】**

```
.vtop { vertical-align: top }
.vmid { vertical-align: middle }
.vbtm { vertical-align: bottom }
em {
  font-style: normal;
  font-weight: bold;
  color: #ff6600;
  background: #ffffff
}
td {
  width: 6em;
  height: 6em;
  text-align: center
}
```

**【HTML】**

```
<p>
vertical-align:
<img class="vmid" src="leaf.gif" width="80" height="80" alt="">
<em>middle</em>
</p>
<p>
vertical-align:
<img src="leaf.gif" width="80" height="80" alt="">
<em class="vtop">top</em>
<em class="vbtm">bottom</em>
</p>

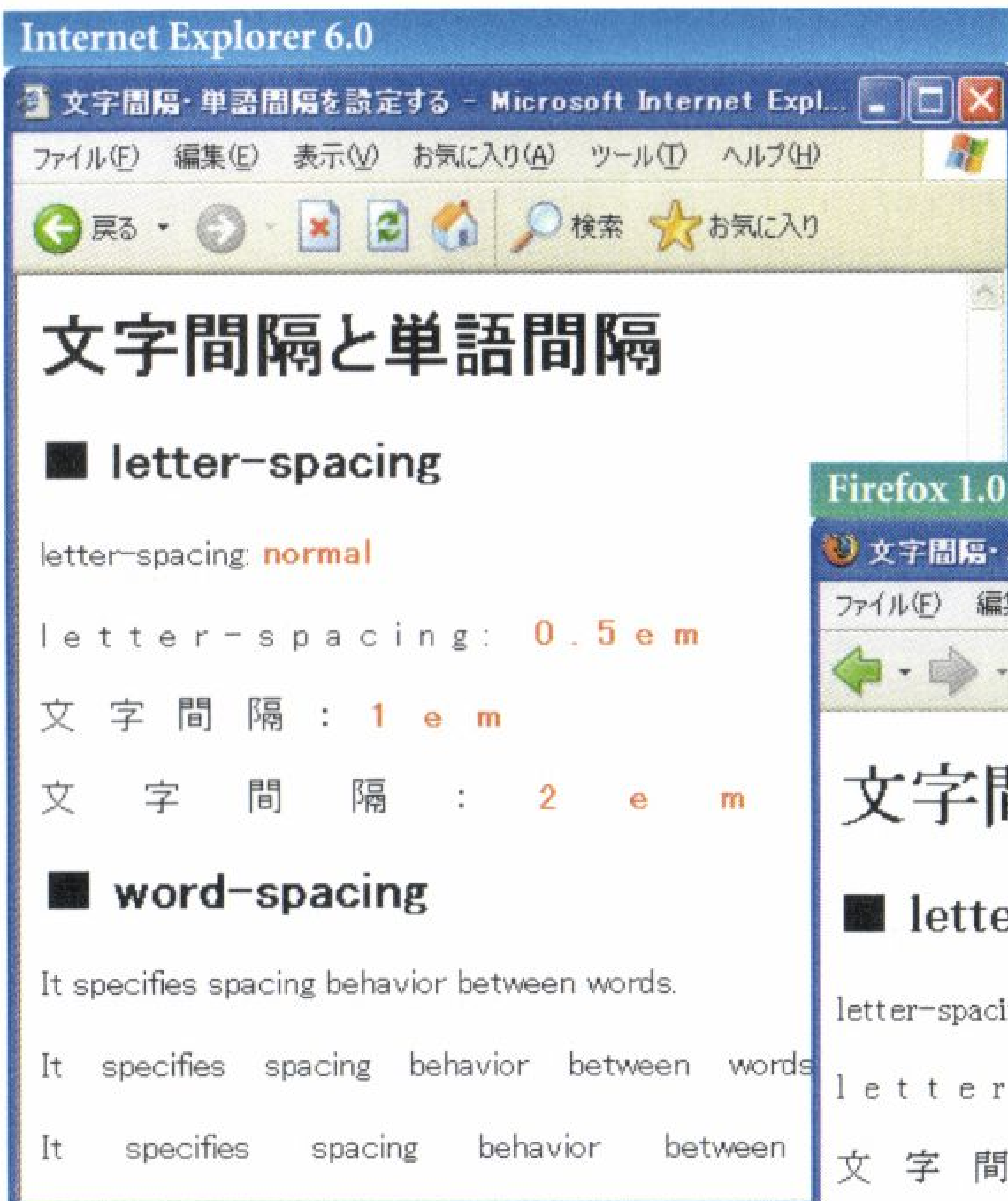
<table border="3">
<tr>
<td class="vtop"><em>top</em></td>
<td class="vmid"><em>middle</em></td>
<td class="vbtm"><em>bottom</em></td>
</tr>
</table>
```

# 文字間隔・単語間隔を設定する

**letter-spacing:** *文字間隔*
**word-spacing:** *単語間隔*

文字間隔・単語間隔　　normal・単位付きの数値

Internet Explorer 6.0




Firefox 1.0




letter-spacing プロパティは文字と文字の間隔を、word-spacing プロパティは単語と単語の間隔を設定します。
数値を指定した場合は、標準の値に対してプラスされます(マイナスも可)。

## Sample

【CSS】

```
.sample1 { letter-spacing: 0.5em }
.sample2 { letter-spacing: 1em }
.sample3 { letter-spacing: 2em }
.sample4 { word-spacing: 0.8em }
.sample5 { word-spacing: 1.5em }
```

```
em {
  font-style: normal;
  font-weight: bold;
  color: #ff6600;
  background: #ffffff
}
```

【HTML】

```
<h1>文字間隔と単語間隔</h1>

<h2>■ letter-spacing</h2>
<p>letter-spacing: <em>normal</em></p>
<p class="sample1">letter-spacing: <em>0.5em</em></p>
<p class="sample2">文字間隔：<em>1em</em></p>
<p class="sample3">文字間隔：<em>2em</em></p>

<h2>■ word-spacing</h2>
<p>It specifies spacing behavior between words.</p>
<p class="sample4">It specifies spacing behavior between words.</p>
<p class="sample5">It specifies spacing behavior between words.</p>
</p>
```



**コラム**

**単語の途中にスペースを入れてはいけない？**

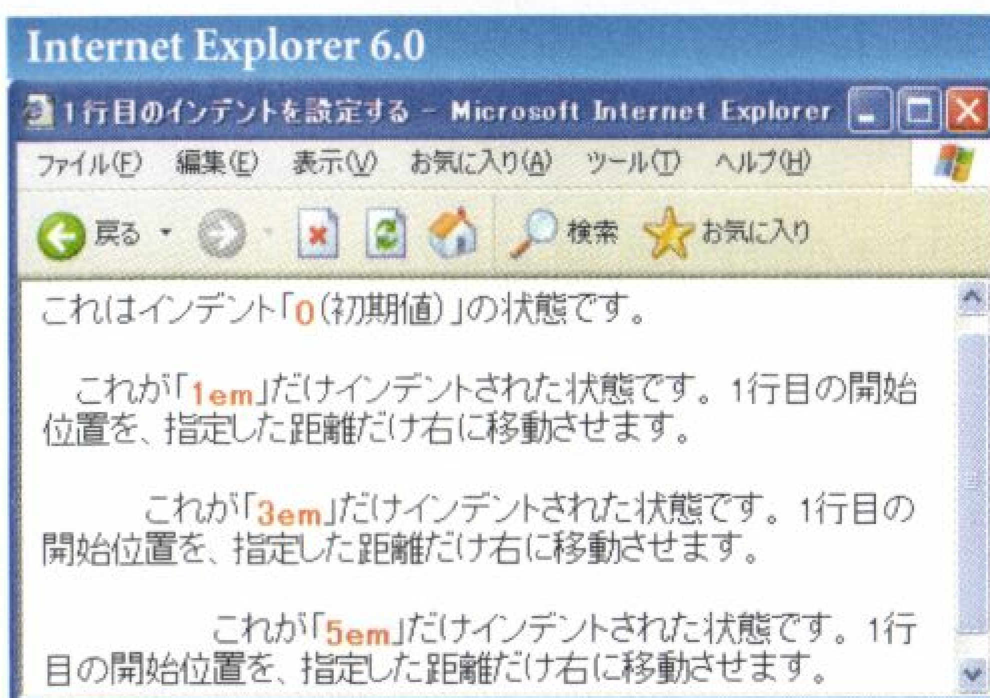
ワープロなどの文書では、「日　時」「場　所」のようにひとつの単語の途中に全角スペースを入れて表示幅を調整することがあります。しかし、ホームページで同様のことを行うと、音声ブラウザでは正しく読み上げられなくなる可能性が高くなります。たとえば、IBMのホームページ・リーダーの場合は、「日　時」は「ひ・とき」、「場　所」は「ば・ところ」のように読み上げられてしまいます。単語の途中にスペースを入れると、ページ内のテキストを検索する場合などにも不都合が生じますので、文字の間隔を空けたい場合にはスタイルシートで調整するようにしましょう。

# 1行目のインデントを設定する

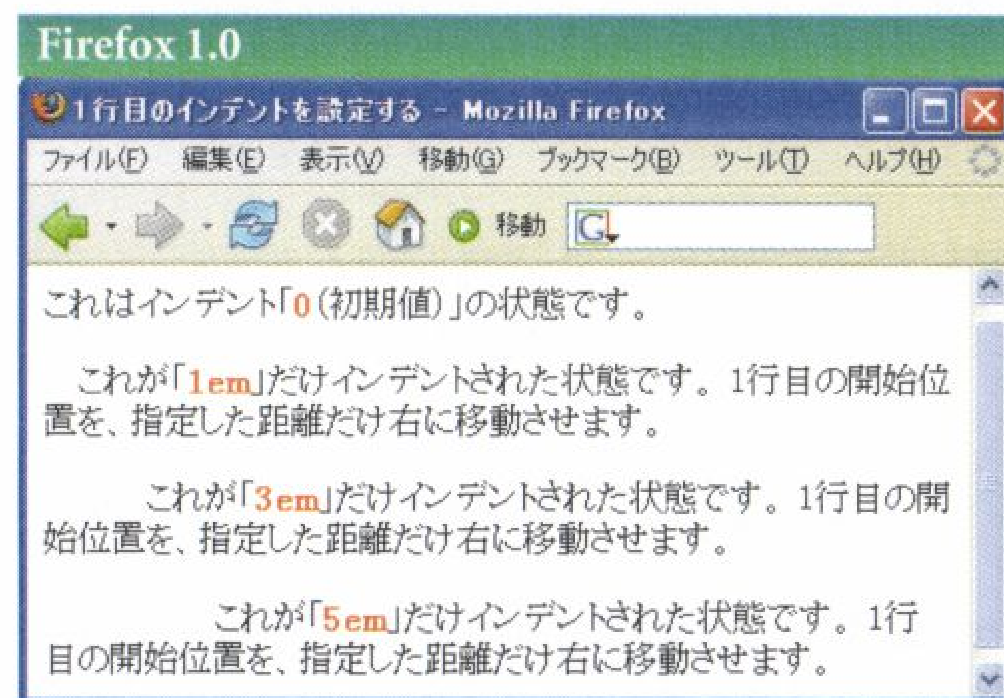
## text-indent: インデント

インデント　単位付きの数値・%

Internet Explorer 6.0



Firefox 1.0



text-indentプロパティは、ブロックレベル要素に指定して、そこに含まれるテキストの先頭のインデントを設定します。
インデントには、マイナスの値を設定することもできます。

## Sample

【CSS】

```
#sample1 { text-indent: 1em }
#sample2 { text-indent: 3em }
#sample3 { text-indent: 5em }
em {
  font-style: normal;
  font-weight: bold;
  color: #ff6600;
  background: #ffffff
}
```

【HTML】

```
<p>これはインデント「<em>0</em>(初期値)」の状態です。</p>
<p id="sample1">
これが「<em>1em</em>」だけインデントされた状態です。
1行目の開始位置を、指定した距離だけ右に移動させます。
</p>
<p id="sample2">
これが「<em>3em</em>」だけインデントされた状態です。
1行目の開始位置を、指定した距離だけ右に移動させます。
</p>
<p id="sample3">
これが「<em>5em</em>」だけインデントされた状態です。
1行目の開始位置を、指定した距離だけ右に移動させます。
</p>
```

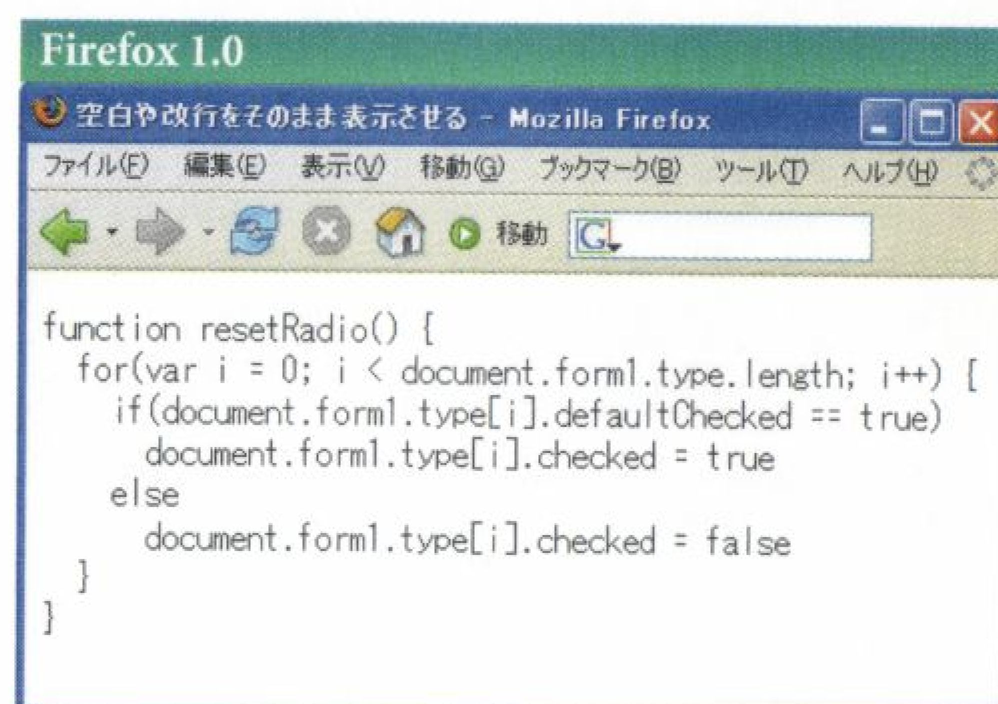
# 空白や改行をそのまま表示させる

## white-space: pre

Internet Explorer 6.0




Firefox 1.0




「white-space: pre」は、連続する半角スペースやタブ、改行を入力されている通りにそのまま表示させる指定です。
Windows版のInternet Explorerの場合、バージョン6.0の標準モードでは利用できますが、互換モードやそれ以前のバージョンでは利用できません。

### Sample

**【CSS】**

```
code { white-space: pre }
```

**【HTML】**

```
<p>
<code>
function resetRadio() {
  for(var i = 0; i &lt; document.form1.type.length; i++) {
    if(document.form1.type[i].defaultChecked == true)
      document.form1.type[i].checked = true
    else
      document.form1.type[i].checked = false
  }
}
</code>
</p>
```

 HTML ：TIPS「新しいブラウザは<!DOCTYPE>の書き方で表示が変わる！」(P.9)

# 改行しないで表示させる

## white-space: nowrap







「white-space: nowrap」は、連続する半角スペースやタブ、改行をひとつの半角スペースに変換して、改行せずに表示させる指定です。

### Sample

**【CSS】**

```
.sample1 { white-space: nowrap }
```

**【HTML】**

```
<h1>改行の禁止</h1>

<h2>■ 普通の状態</h2>
<p>
white-spaceプロパティに「nowrap」を指定すると、その範囲のテキストは改行されなくなります。中に含まれている連続する半角スペースやタブ、改行コードはひとつの半角スペースに変換されます。
</p>

<h2>■ white-space: nowrap</h2>
<p class="sample1">
white-spaceプロパティに「nowrap」を指定すると、その範囲のテキストは改行されなくなります。中に含まれている連続する半角スペースやタブ、改行コードはひとつの半角スペースに変換されます。
</p>
```

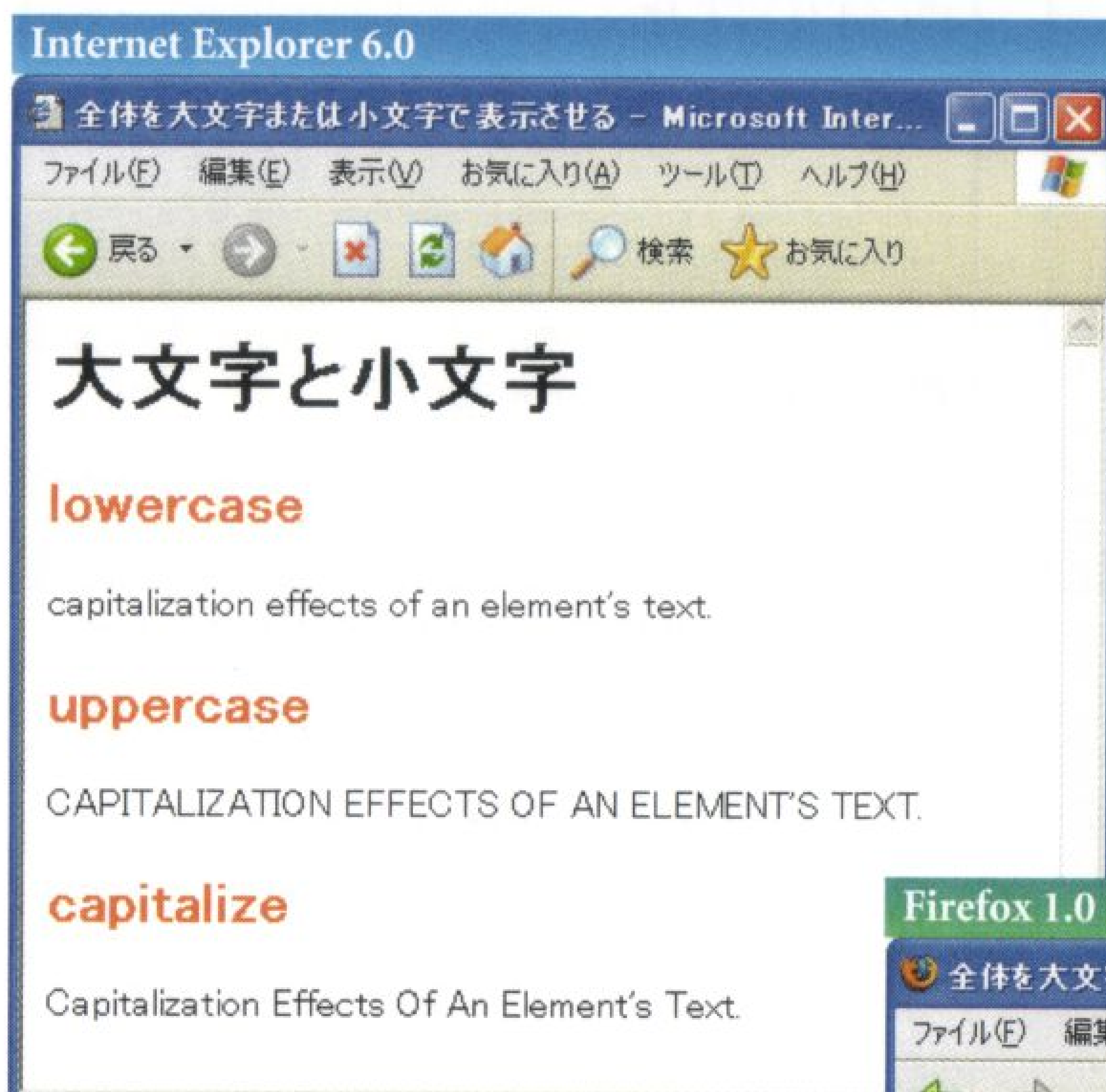
IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 全体を大文字または小文字で表示させる

## text-transform: 大文字・小文字の指定

| 【大文字・小文字の指定】 | |
|---|---|
| uppercase | すべての文字を大文字で表示 |
| lowercase | すべての文字を小文字で表示 |
| capitalize | 各単語の先頭の文字だけ大文字で表示 |

Internet Explorer 6.0




Firefox 1.0




text-transform プロパティは、全体を大文字や小文字で表示させたり、各単語の先頭の文字だけを大文字で表示させることのできる指定です。
日本語の部分は特に変化しません。

テキストを設定する

## Sample

**【CSS】**

```
#sample1 { text-transform: lowercase }
#sample2 { text-transform: uppercase }
#sample3 { text-transform: capitalize }
h2 {
  color: #ff6600;
  background: #ffffff
}
```

**【HTML】**

```
<h1>大文字と小文字</h1>

<h2>lowercase</h2>
<p id="sample1">
CAPITALIZATION EFFECTS OF AN ELEMENT'S TEXT.
</p>

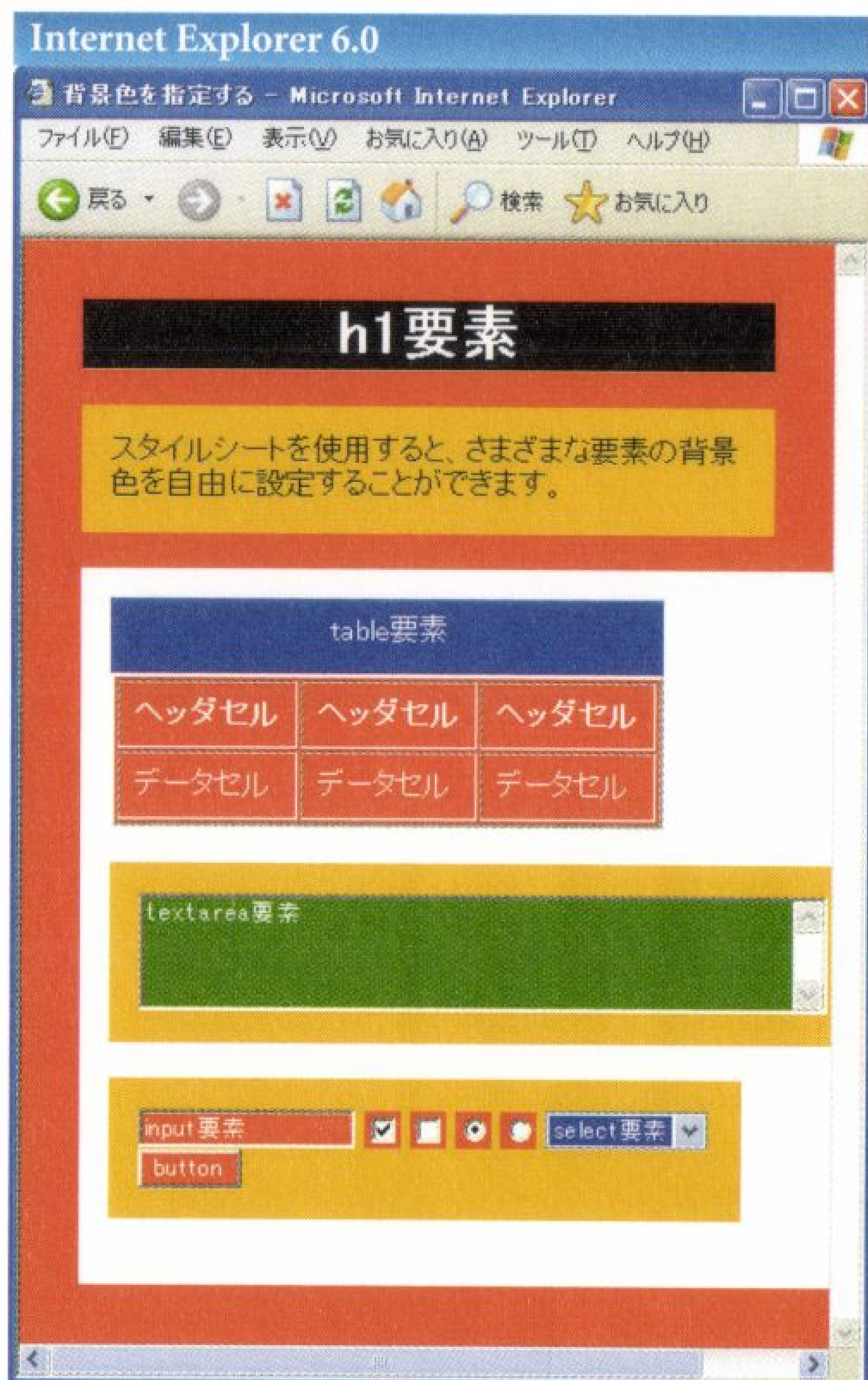
<h2>uppercase</h2>
<p id="sample2">
capitalization effects of an element's text.
</p>

<h2>capitalize</h2>
<p id="sample3">
capitalization effects of an element's text.
</p>
```

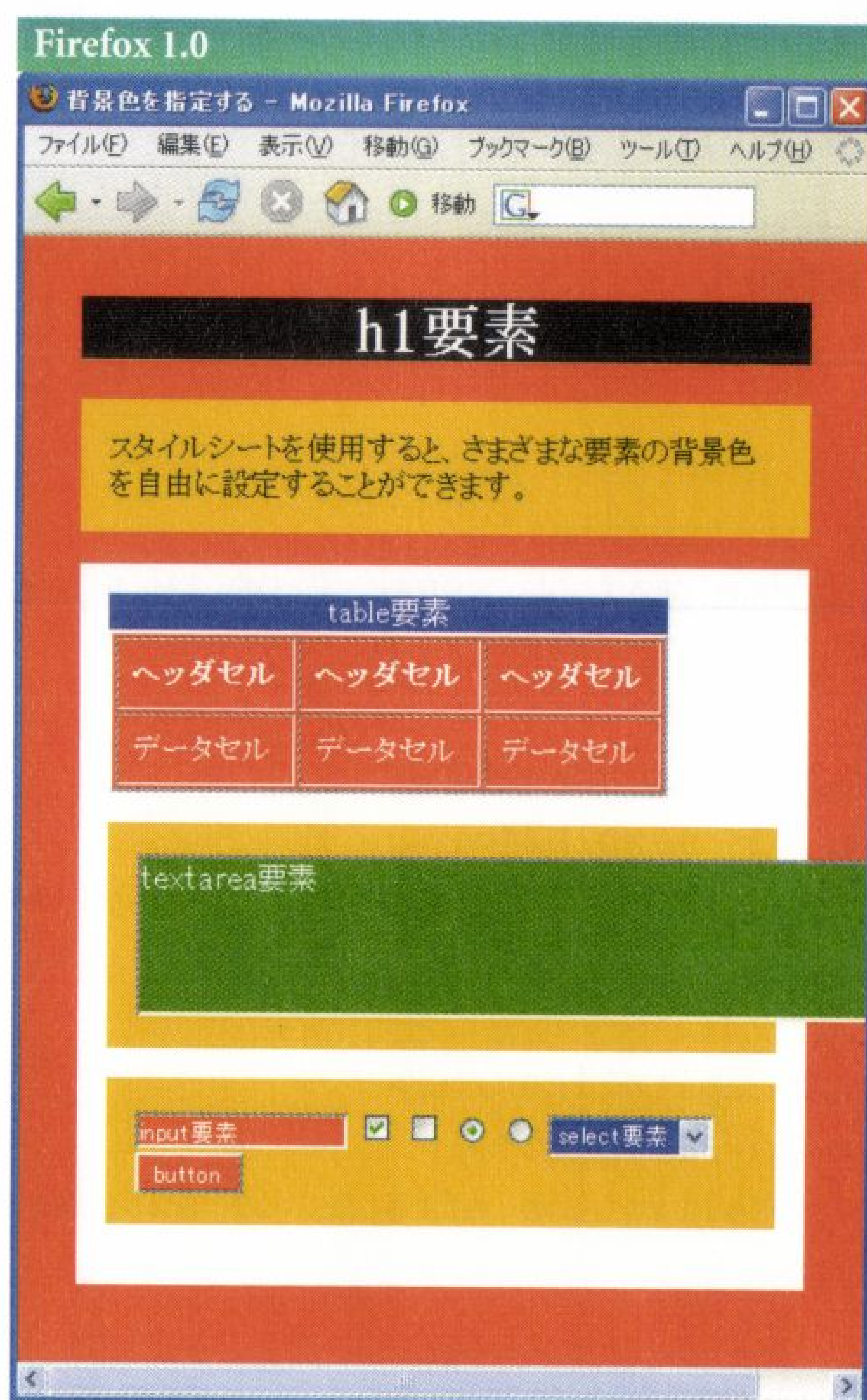
# 背景色を指定する

## background-color: 色指定

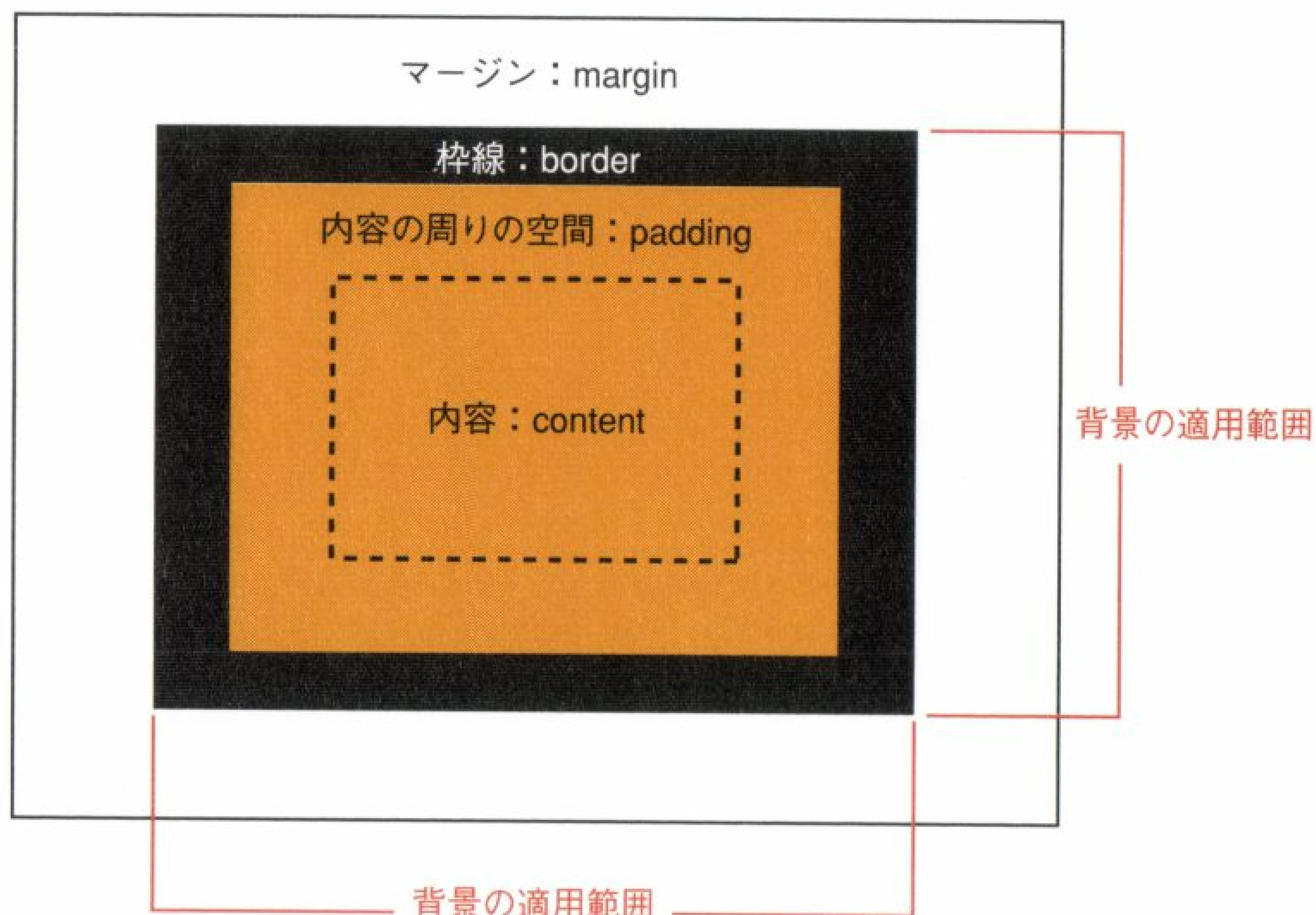
Internet Explorer 6.0




Firefox 1.0




ボックスと背景の適用範囲



background-colorプロパティは、背景の色を指定します。
色の値として「transparent」を指定すると、背景が透明になって下の背景が透けて見えるようになります。ここで設定した色はボックスのマージン（常に透明）には適用されません。
背景色だけを指定していると、ユーザーが設定画面や独自のスタイルシートで文字色を指定している場合などに、文字が読みにくくなってしまう場合があります。そのようなことを避けるために、背景色を指定する場合には文字色も同時に指定するようにしてください。

## Sample

**【CSS】**

```
body {
  margin: 2em;
  color: #ffffff;
  background-color: #ff3300
}
h1 {
  text-align: center;
  color: #ffffff;
  background-color: #000000
}
p {
  padding: 1em;
  color: #000000;
  background-color: #ffcc00
}
div {
  padding: 1em;
  color: #000000;
  background-color: #ffffff
}
table, input {
  color: #ffffff;
  background-color: #ff3300
}
caption, select {
  color: #ffffff;
  background-color: #333399
}
textarea {
  color: #ffffff;
  background-color: #339933
}
```

**【HTML】**

```
<h1>h1要素</h1>

<p>
スタイルシートを使用すると、さまざまな要素の背景色を自由に設定することができます。
</p>

<div>
<table border="2" cellpadding="8">
<caption>table要素</caption>
<tr><th>ヘッダセル</th><th>ヘッダセル</th><th>ヘッダセル</th></tr>
<tr><td>データセル</td><td>データセル</td><td>データセル</td></tr>
</table>
<p>
<textarea rows="4" cols="50">textarea要素</textarea>
</p>
<p>
<input type="text" value="input要素">
<input type="checkbox" name="chk" checked>
<input type="checkbox" name="chk">
<input type="radio" name="rdo" checked>
<input type="radio" name="rdo">
<select>
  <option>select要素</option>
  <option>option要素</option>
</select>
<input type="button" value="button">
</p>
</div>
```

色指定：「CSSについて」の「色の指定方法」(P.180)
color：「フォント」の「文字色を指定する」(P.201)
色指定：巻末付録「カラーチャート1～3」(巻末)

# 背景画像を指定する

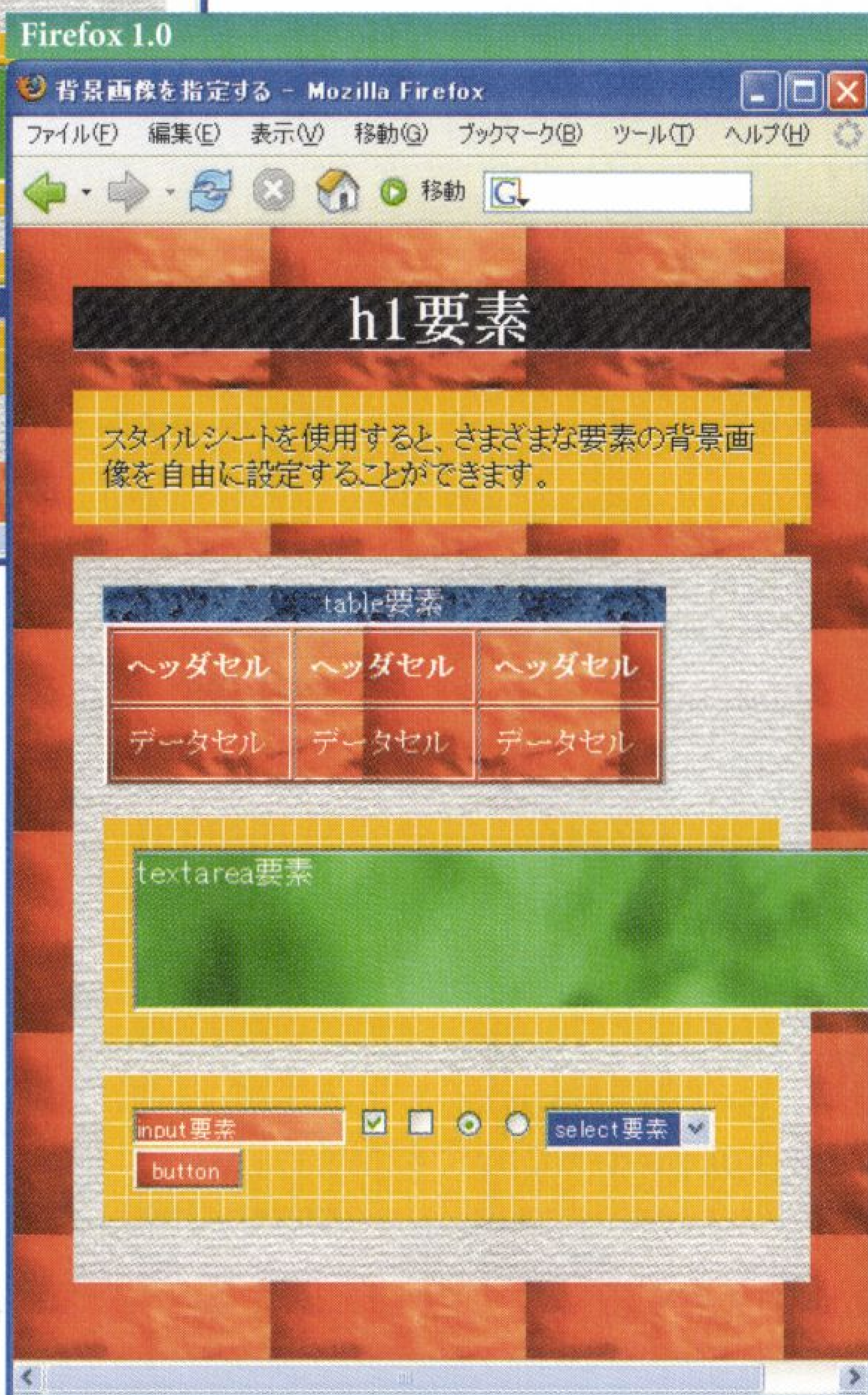
## background-image: url(*URL*)

| | |
|---|---|
| URL | 画像のURL |

Internet Explorer 6.0




Firefox 1.0



background-image プロパティは、背景の画像を指定します。
背景色の場合と同様に、ここで設定した画像はボックスのマージン(常に透明)には適用されません。環境によっては、背景画像が表示されない場合もありますので、背景色(文字色)も同時に指定しておくようにしてください。

## Sample

**【CSS】**

```
body {
  margin: 2em;
  color: #ffffff;
  background-color: #ff3300;
  background-image: url(red.jpg)
}
h1 {
  text-align: center;
  color: #ffffff;
  background-color: #000000;
  background-image: url(black.jpg)
}
p {
  padding: 1em;
  color: #000000;
  background-color: #ffcc00;
  background-image: url(yellow.gif)
}
div {
  padding: 1em;
  color: #000000;
  background-color: #ffffff;
  background-image: url(white.jpg)
}
table, input {
  color: #ffffff;
  background-color: #ff3300;
  background-image: url(red.jpg)
}
caption, select {
  color: #ffffff;
  background-color: #333399;
  background-image: url(blue.gif)
}
textarea {
  color: #ffffff;
  background-color: #339933;
  background-image: url(green.jpg)
}
```

【HTML】

```
<h1>h1要素</h1>

<p>
スタイルシートを使用すると、さまざまな要素の背景画像を自由に設定することができます。
</p>

<div>
<table border="2" cellpadding="8">
<caption>table要素</caption>
<tr><th>ヘッダセル</th><th>ヘッダセル</th><th>ヘッダセル</th></tr>
<tr><td>データセル</td><td>データセル</td><td>データセル</td></tr>
</table>
<p>
<textarea rows="4" cols="50">textarea要素</textarea>
</p>
<p>
<input type="text" value="input要素">
<input type="checkbox" name="chk" checked>
<input type="checkbox" name="chk">
<input type="radio" name="rdo" checked>
<input type="radio" name="rdo">
<select>
  <option>select要素</option>
  <option>option要素</option>
</select>
<input type="button" value="button">
</p>
</div>
```

CSS

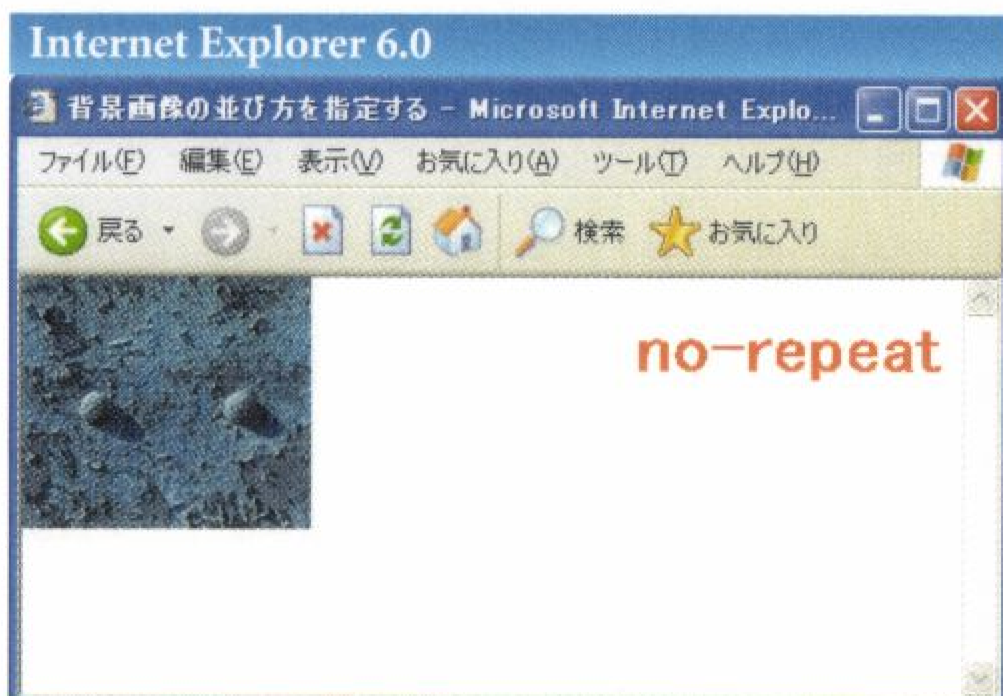
# 背景画像の並び方を指定する

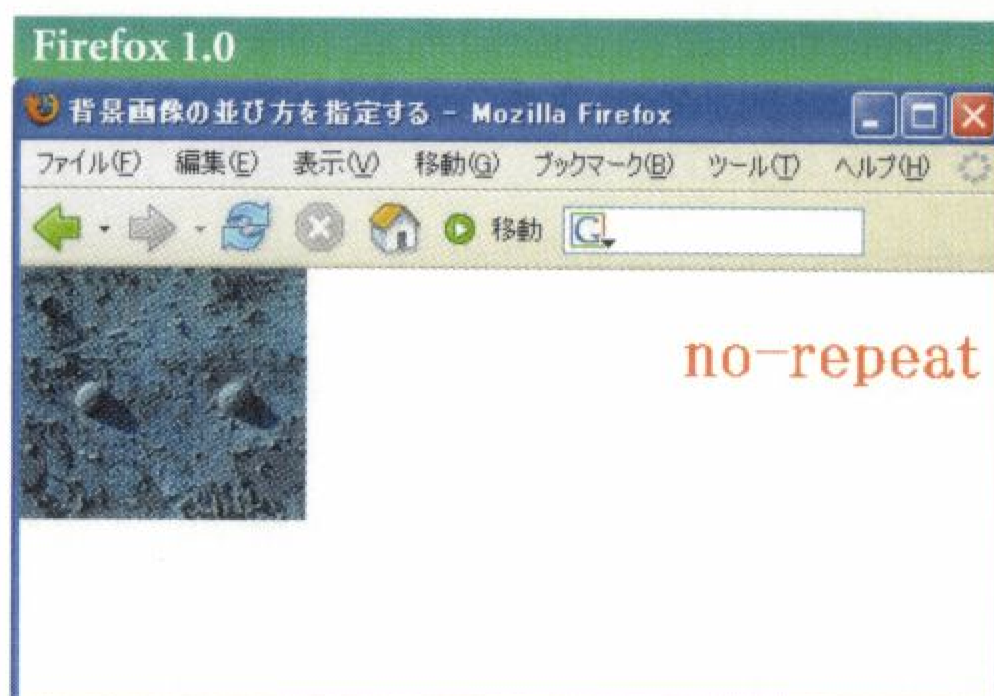
**background-repeat:** *並び方*

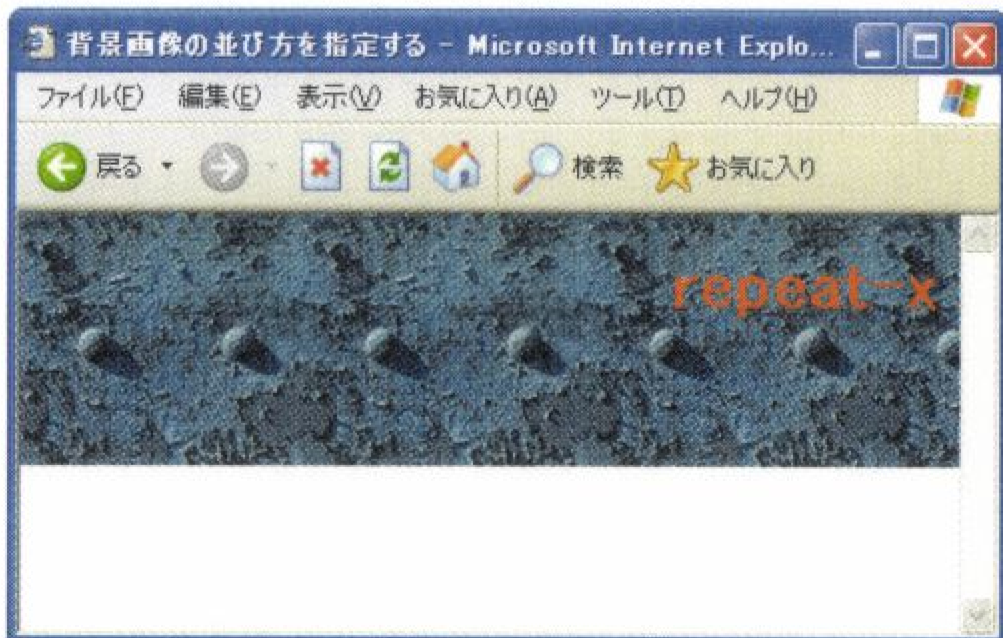
| 【並び方】 | |
|---|---|
| repeat | 縦横にタイル状に繰り返して表示（初期値） |
| repeat-x | 横方向にのみ繰り返して表示 |
| repeat-y | 縦方向にのみ繰り返して表示 |
| no-repeat | 繰り返さずにひとつだけ表示 |

**Internet Explorer 6.0**

**Firefox 1.0**

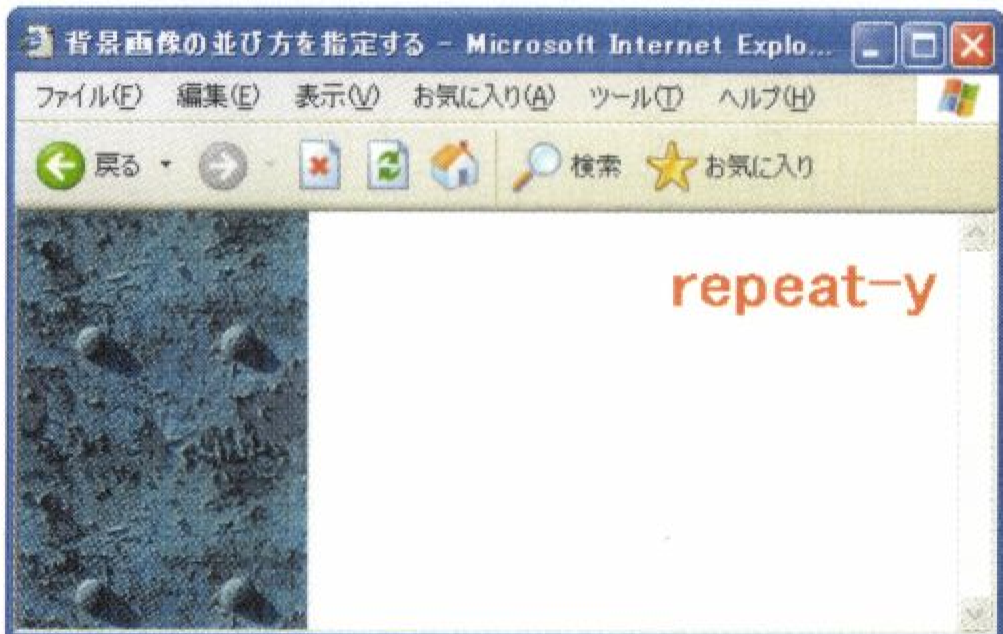


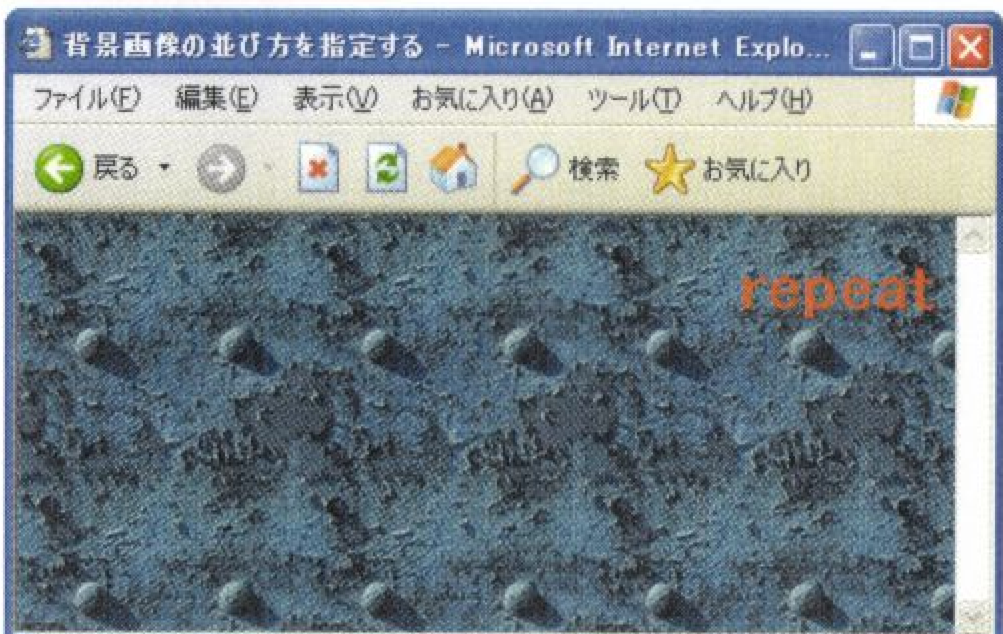


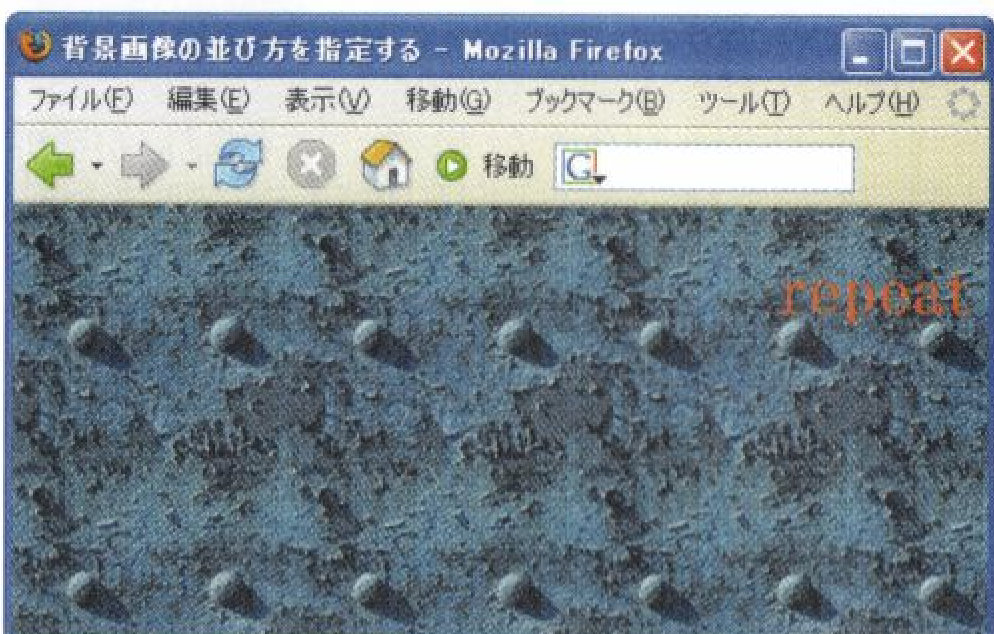










background-repeat プロパティは、背景画像が指定された場合の、画像の並び方を設定します。

**Sample**

**【CSS】**

```
body {
  color: #ff6600;
  background-color: #ffffff;
  background-image: url(back.gif);
  background-repeat: no-repeat
}
h1 { text-align: right }
```

**【HTML】**

```
<h1>no-repeat</h1>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 背景画像の表示位置を指定する

**background-position: 表示位置**

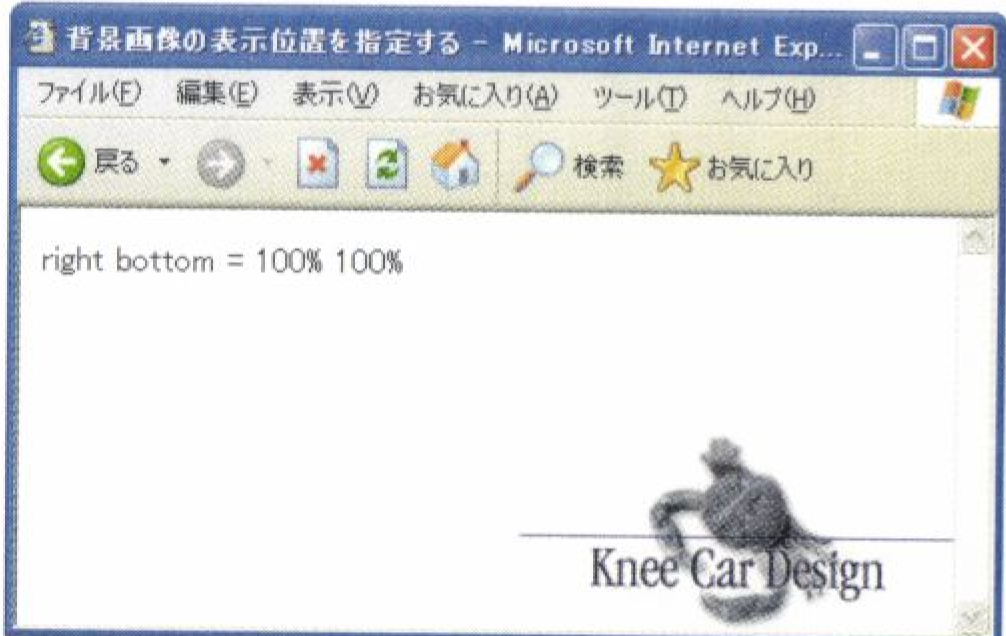
| 【表示位置】 | |
|---|---|
| 横位置 縦位置 | 左上を起点とした単位付きの数値または% |
| 横位置 縦位置 | left・right・center・top・bottom |

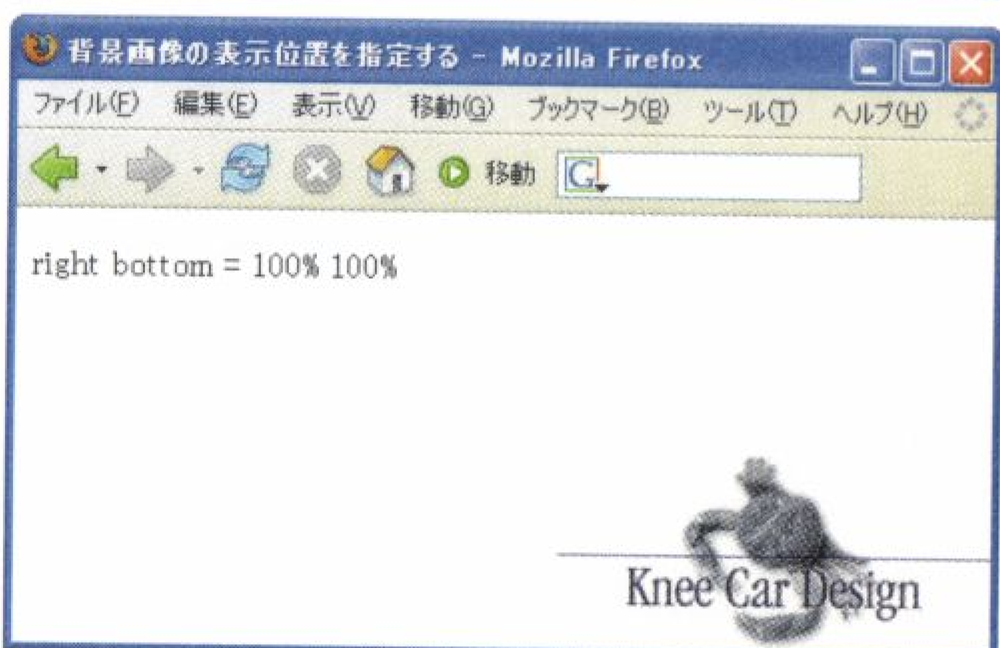
Internet Explorer 6.0




Firefox 1.0




background-position プロパティは、背景画像が指定された場合の、画像の表示位置を設定します。
単位付きの数値と%による指定の場合は、横位置・縦位置の順に半角スペースで区切って指定します。値がひとつしか指定されなかった場合は、横位置が指定されたことになり、その場合の縦位置は「50%」の位置になります。単位付きの数値の場合は、領域の左上から画像の左上までの距離を指定します。%による指定の場合は領域の指定した割合の位置に、画像の同じ割合の位置が合わせて表示されます。これらは混在させて指定することが可能です。位置を示すキーワード(left や top など)は、それぞれ left と top は「0%」、center は「50%」、right と bottom は「100%」を指定した場合と同じ結果になります。これらは順不同で指定することができ、ひとつしか指定しなかった場合はもう一方が center になります。

## Sample

### 【CSS】

```
body {
  color: #000000;
  background-color: #ffffff;
  background-image: url(logo.gif);
  background-repeat: no-repeat;
  background-position: center center
}
```

### 【HTML】

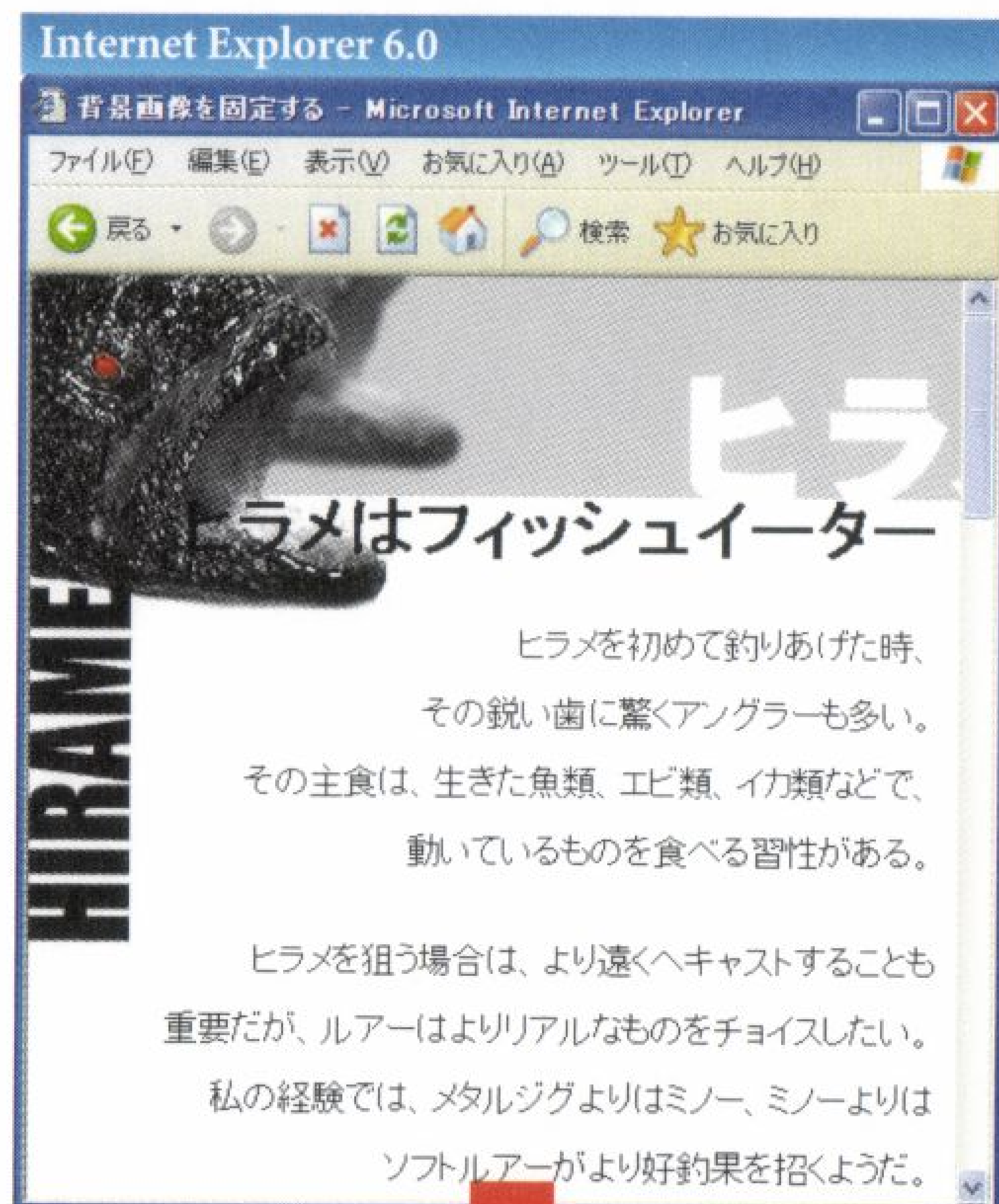
```
<p>center center = 50% 50%</p>
```

# 背景画像を固定する

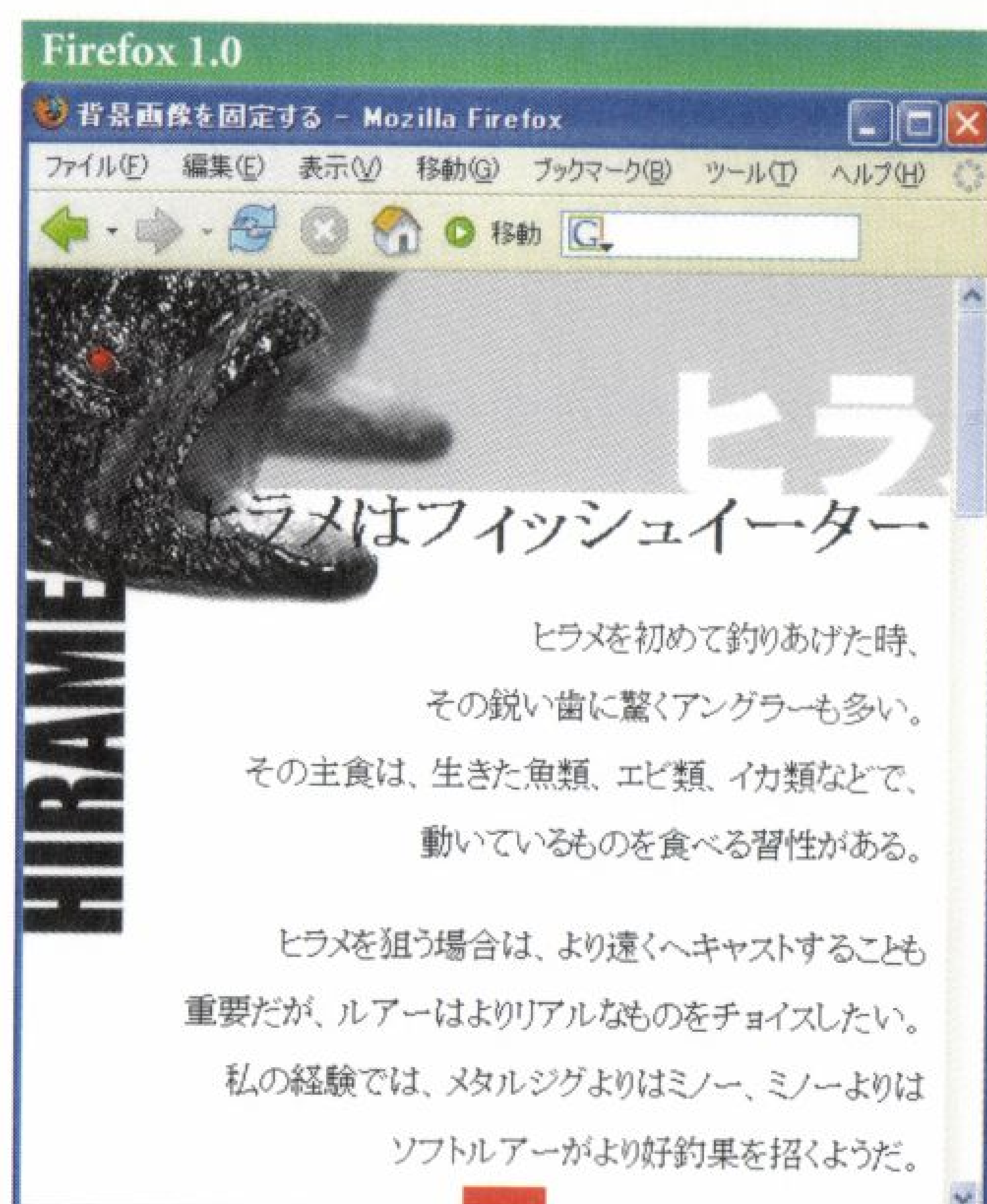
## background-attachment: 固定するかどうか

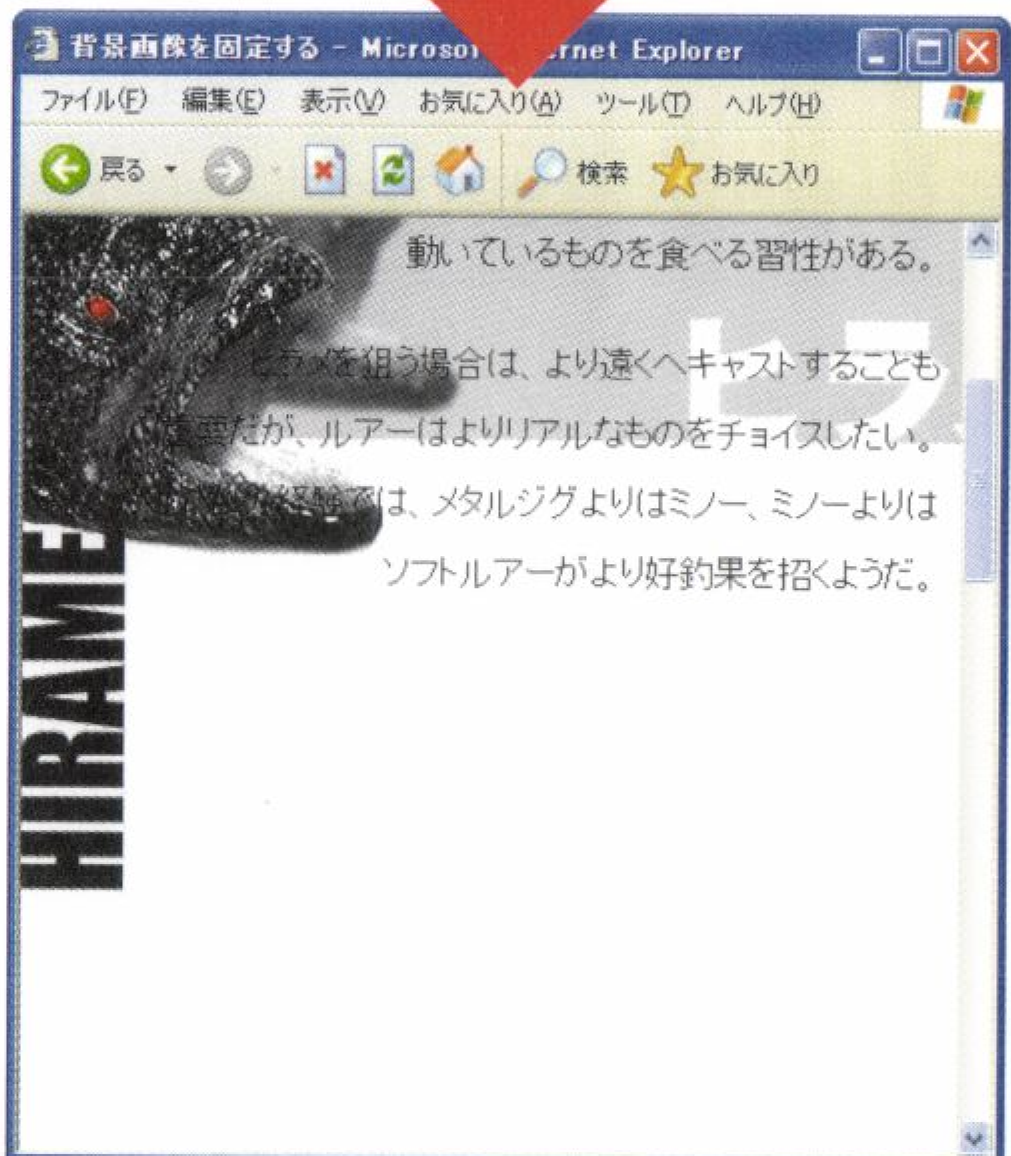
| 【固定するかどうか】 | |
|---|---|
| fixed | 背景画像の位置を固定する |
| scroll | 背景画像を他の内容と共にスクロールさせる（初期値） |

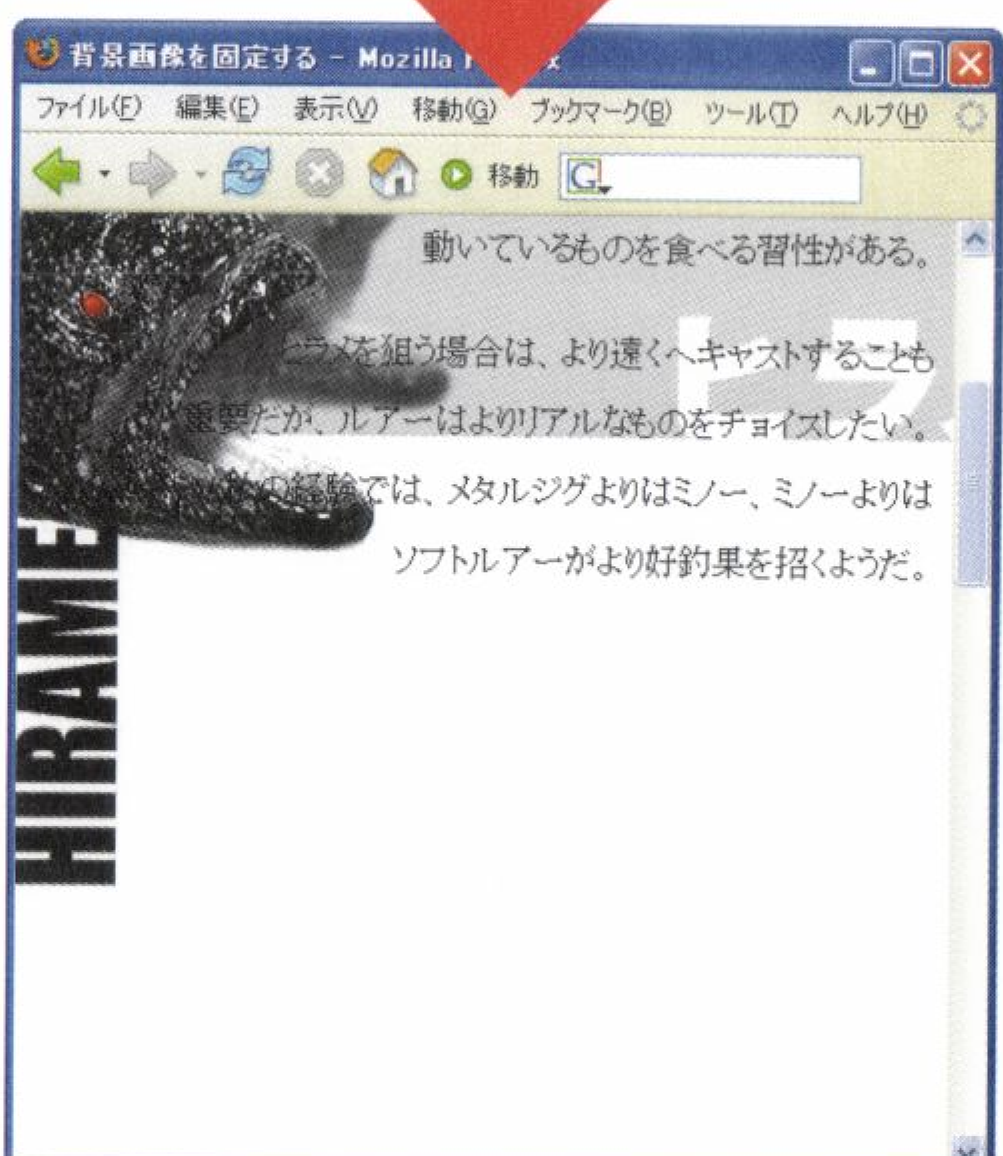
Internet Explorer 6.0




Firefox 1.0




background-attachment プロパティは、背景画像が指定された場合に、その画像を（ウィンドウに対して）その位置に固定するか、他の内容と共にスクロールさせるかを指定します。

## Sample

**【CSS】**

```
body {
  color: #000000;
  background-color: #ffffff;
  background-image: url(hirame.jpg);
  background-repeat: no-repeat;
  background-attachment: fixed;
  margin-top: 100px;
  text-align: right
}
p { line-height: 2em }
```

**【HTML】**

```
<h1>ヒラメはフィッシュイーター</h1>
<p>
ヒラメを初めて釣りあげた時、<br>
その鋭い歯に驚くアングラーも多い。<br>
その主食は、生きた魚類、エビ類、イカ類などで、<br>
動いているものを食べる習性がある。
</p>
<p>
ヒラメを狙う場合は、より遠くへキャストすることも<br>
重要だが、ルアーはよりリアルなものをチョイスしたい。<br>
私の経験では、メタルジグよりはミノー、ミノーよりは<br>
ソフトルアーがより好釣果を招くようだ。
</p>
```

# 背景関係をまとめて指定する

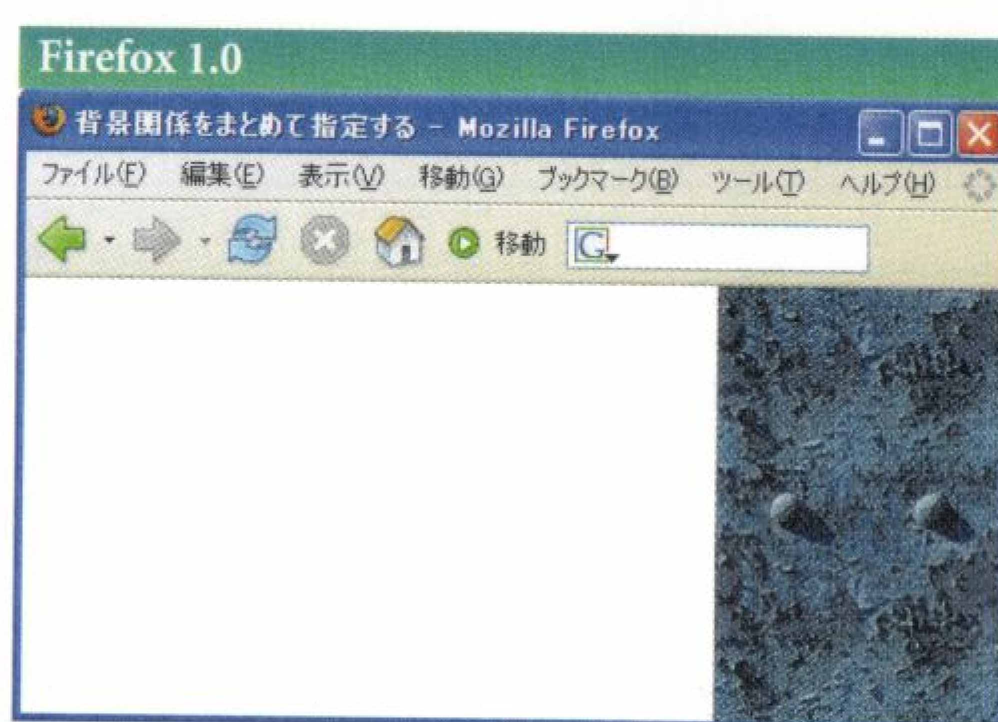
## background: 背景関連のプロパティの値

| 【背景関連のプロパティの値】 |
|---|
| background-color (P.224) で指定できる値 |
| background-image (P.227) で指定できる値 |
| background-repeat (P.230) で指定できる値 |
| background-position (P.232) で指定できる値 |
| background-attachment (P.234) で指定できる値 |

Internet Explorer 6.0




Firefox 1.0




backgroundプロパティは、背景関連のプロパティの値をまとめて指定します。必要な値を任意の順序で半角スペースで区切って指定します。

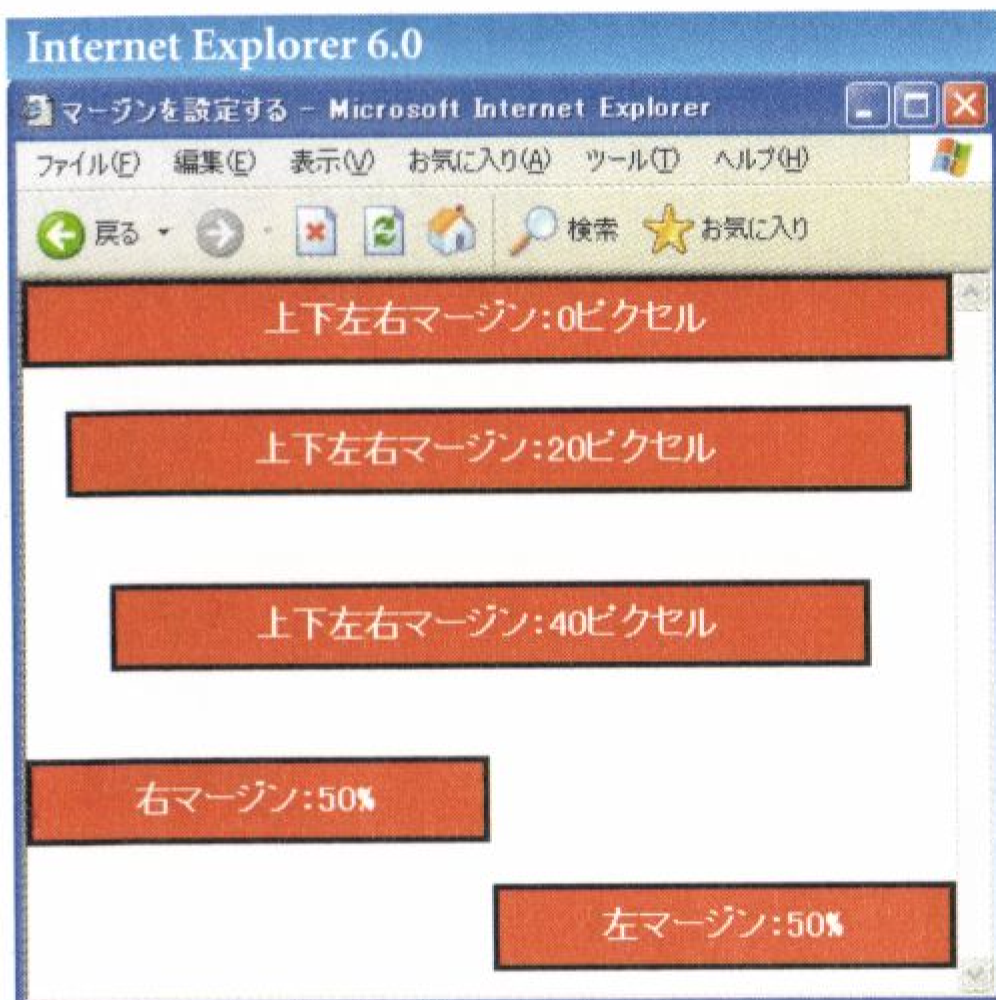
### Sample

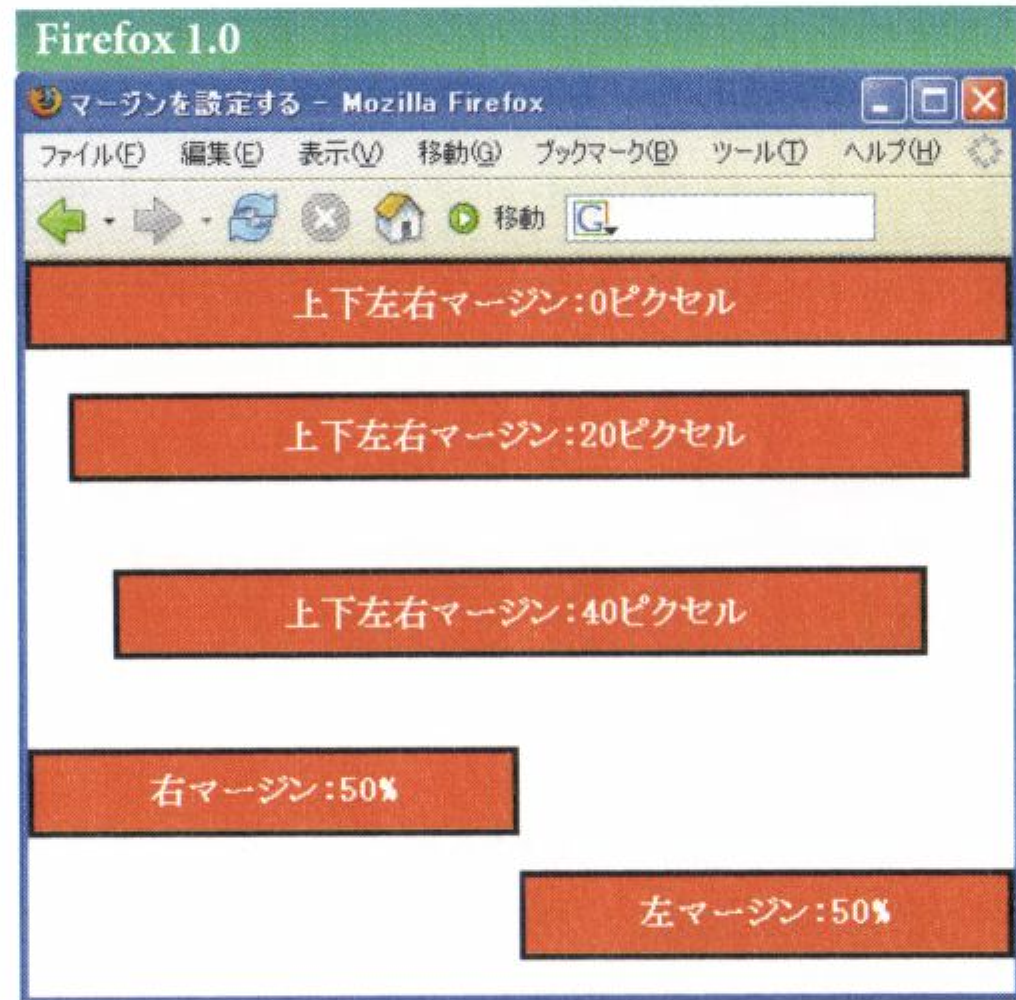
```
body {
  color: #000000;
  background: #ffffff url(back.gif) right repeat-y
}
```

# マージンを設定する

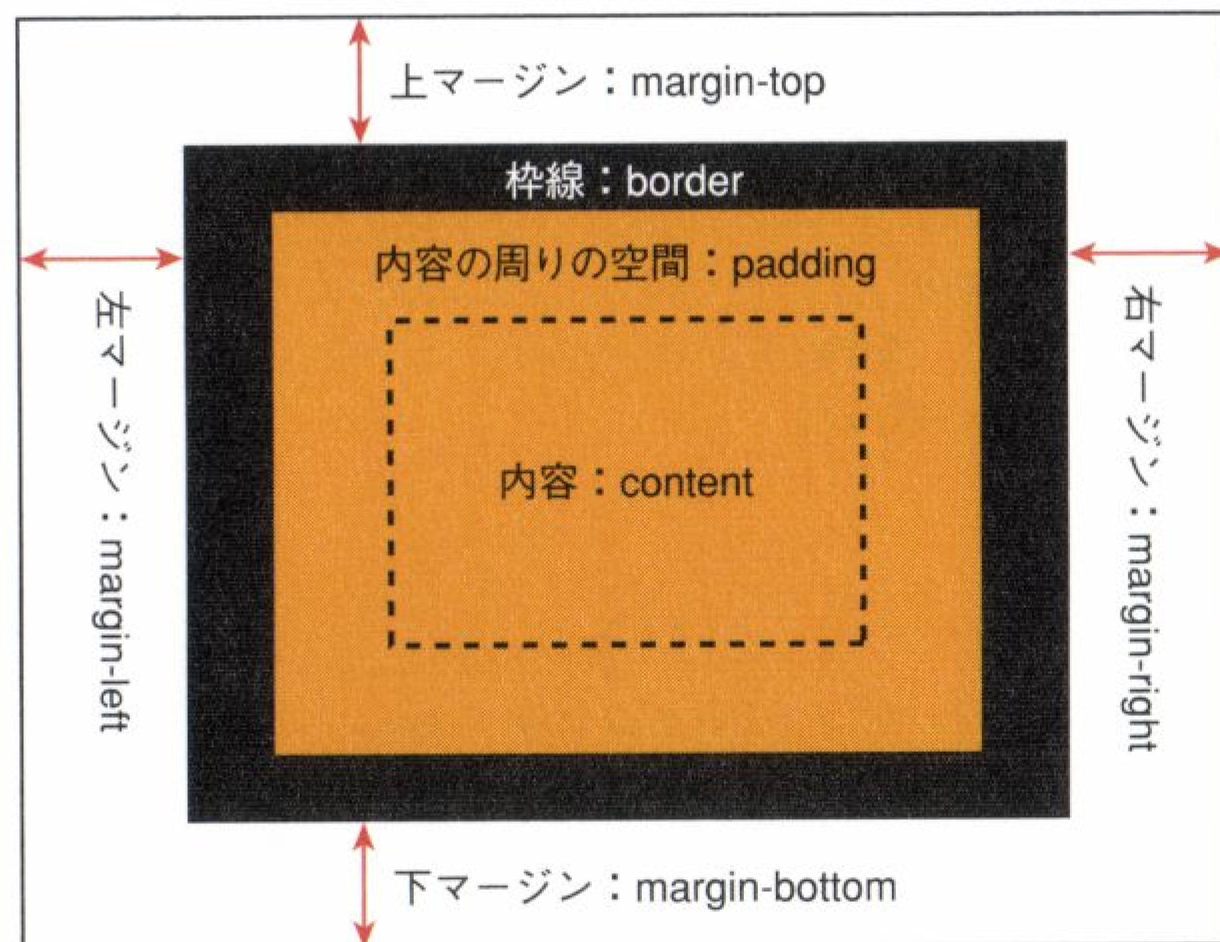
**margin-top:** マージン ←上マージン
**margin-bottom:** マージン ←下マージン
**margin-left:** マージン ←左マージン
**margin-right:** マージン ←右マージン
**margin:** マージン ←上・右・下・左マージン

マージン　単位付きの数値・%・auto




**ボックスの構造**




これらのプロパティは、マージンを設定します。
%で指定した場合には、指定されたボックスを含むボックスの横幅に対する割合となります。上下のマージンについても、高さではなく横幅を参照しますので、注意してください。値として「auto」を指定すると、マージンは状況に応じて自動的に調整され

ます。たとえば、インライン要素の上下左右とブロックレベル要素の上下については0になり、ブロックレベル要素の左右に指定すると両者が同じ値となるためにセンタリングされます。

「margin」を利用すると、上下左右のマージンを一度に設定することができます。その場合、値を半角スペースで区切って指定しますが、与えられた値の個数によって次のようにマージンが設定されます。

| | |
|---|---|
| ・値が1つの場合 | 値1→上下左右 |
| ・値が2つの場合 | 値1→上下　値2→左右 |
| ・値が3つの場合 | 値1→上　値2→左右　値3→下 |
| ・値が4つの場合 | 値1→上　値2→右　値3→下　値4→左 |

なお、上下に隣接するブロックレベル要素同士のマージンは、相殺されて大きい方のマージンだけになります。また、マージン部分は常に透明で色を設定することはできません。

## Sample

**【CSS】**

```
body { margin: 0 }
p {
  text-align: center;
  font-weight: bold;
  padding: 0.5em;
  border: solid 3px #000000;
  color: #ffffff;
  background: #ff3300
}
#sample1 { margin: 0 }
#sample2 { margin: 20px }
#sample3 { margin: 40px }
#sample4 { margin-right: 50% }
#sample5 { margin-left: 50% }
```

**【HTML】**

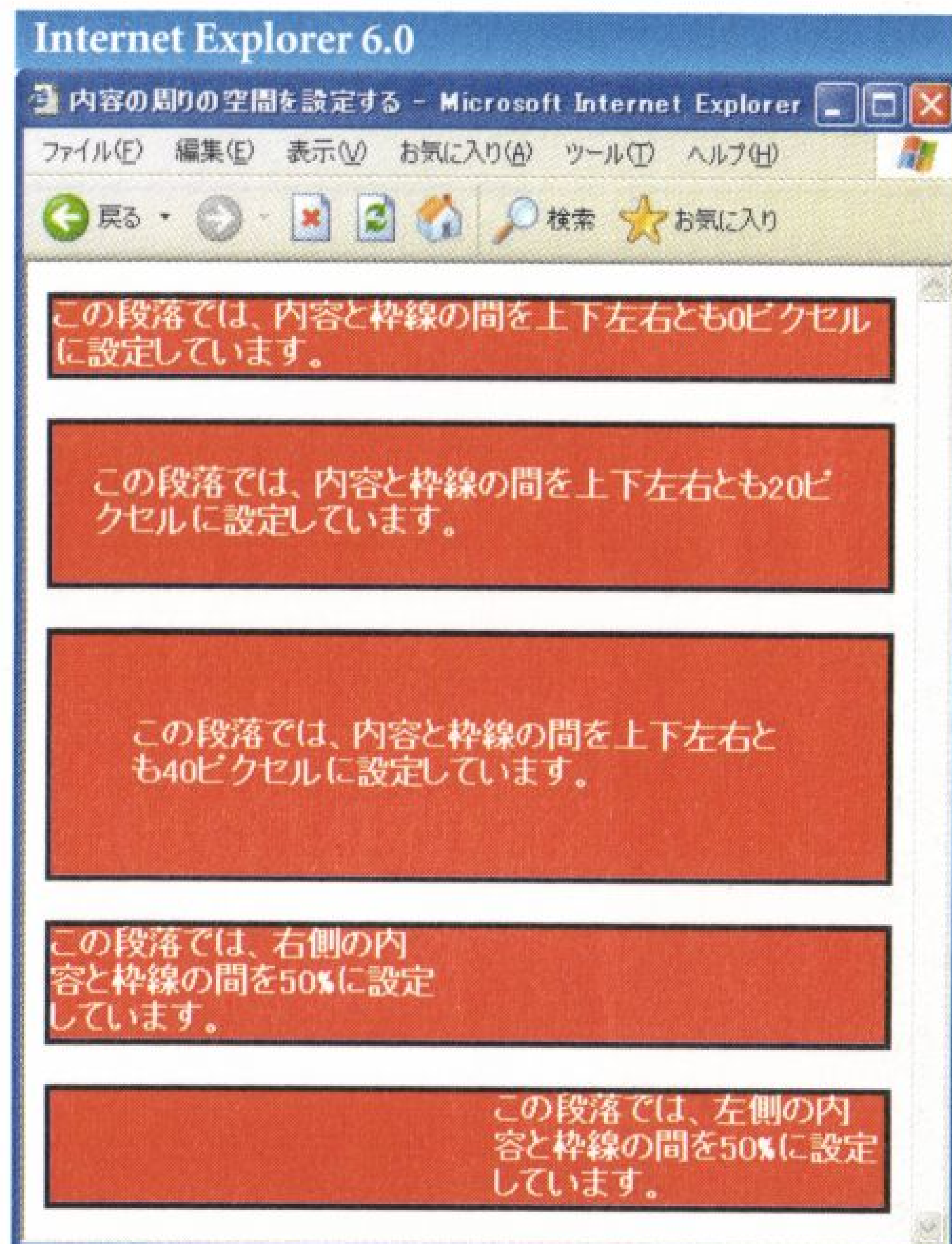
```
<p id="sample1">上下左右マージン：0ピクセル</p>
<p id="sample2">上下左右マージン：20ピクセル</p>
<p id="sample3">上下左右マージン：40ピクセル</p>
<p id="sample4">右マージン：50%</p>
<p id="sample5">左マージン：50%</p>
```

# 内容の周りの空間を設定する

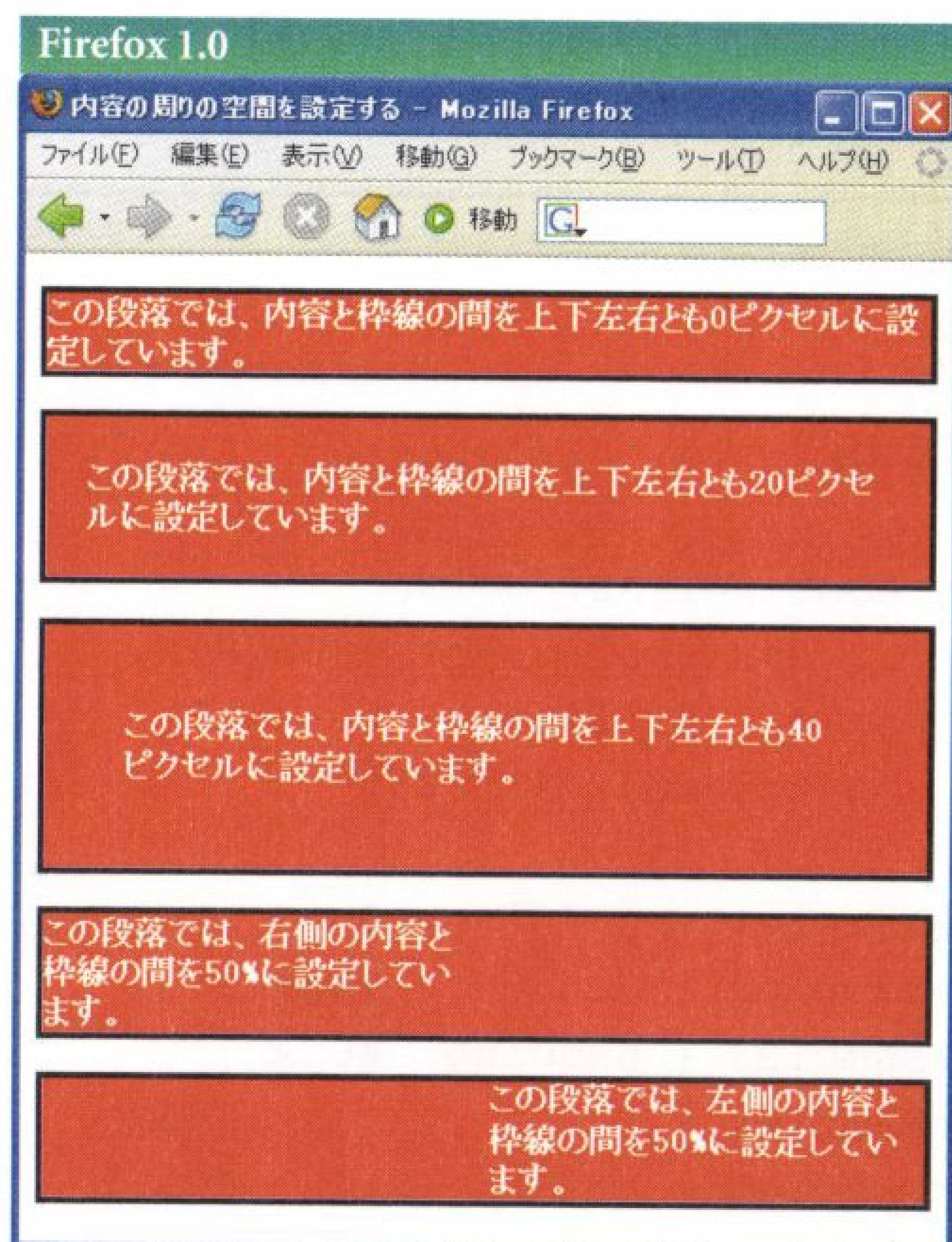
**padding-top:** *幅* ←上の空間
**padding-bottom:** *幅* ←下の空間
**padding-left:** *幅* ←左の空間
**padding-right:** *幅* ←右の空間
**padding:** *幅* ←上・右・下・左の空間

| | |
|---|---|
| 幅 | 単位付きの数値・% |

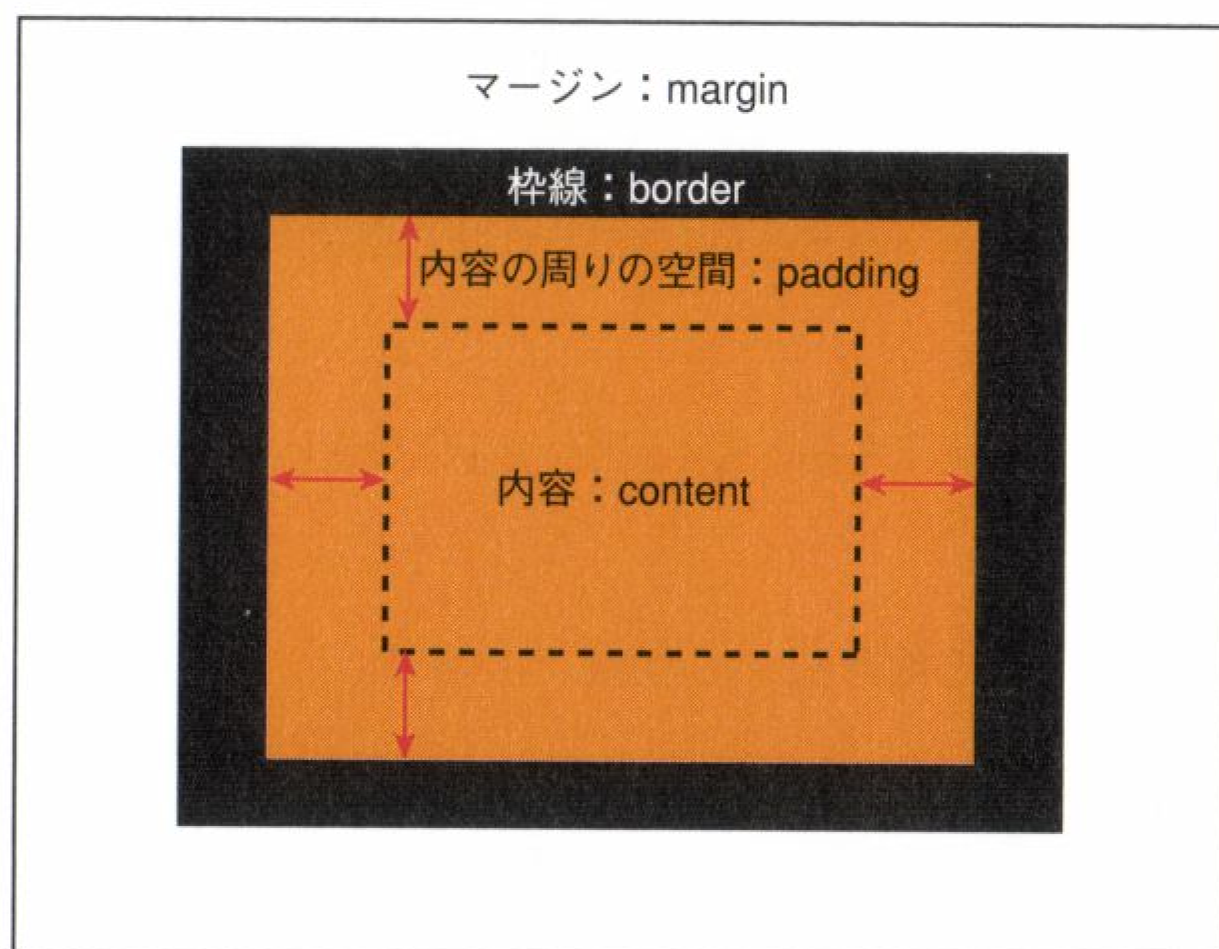
Internet Explorer 6.0




Firefox 1.0




**ボックスの構造**



これらのプロパティは、内容を表示する領域と枠線の間の空間の幅を設定します。%で指定した場合には、指定されたボックスを含んだボックスの横幅に対する割合となります。上下の空間の幅についても、高さではなく横幅を参照しますので、注意してください。

「padding」を利用すると、上下左右の幅を一度に設定することができます。その場合、値を半角スペースで区切って指定しますが、与えられた値の個数によって、次のように幅が設定されます。

| | |
|---|---|
| ・値が1つの場合 | 値1→上下左右 |
| ・値が2つの場合 | 値1→上下　値2→左右 |
| ・値が3つの場合 | 値1→上　値2→左右　値3→下 |
| ・値が4つの場合 | 値1→上　値2→右　値3→下　値4→左 |

## Sample

**【CSS】**

```
p {
  font-weight: bold;
  border: solid 3px #000000;
  color: #ffffff;
  background: #ff3300
}
#sample1 { padding: 0 }
#sample2 { padding: 20px }
#sample3 { padding: 40px }
#sample4 { padding-right: 50% }
#sample5 { padding-left: 50% }
```

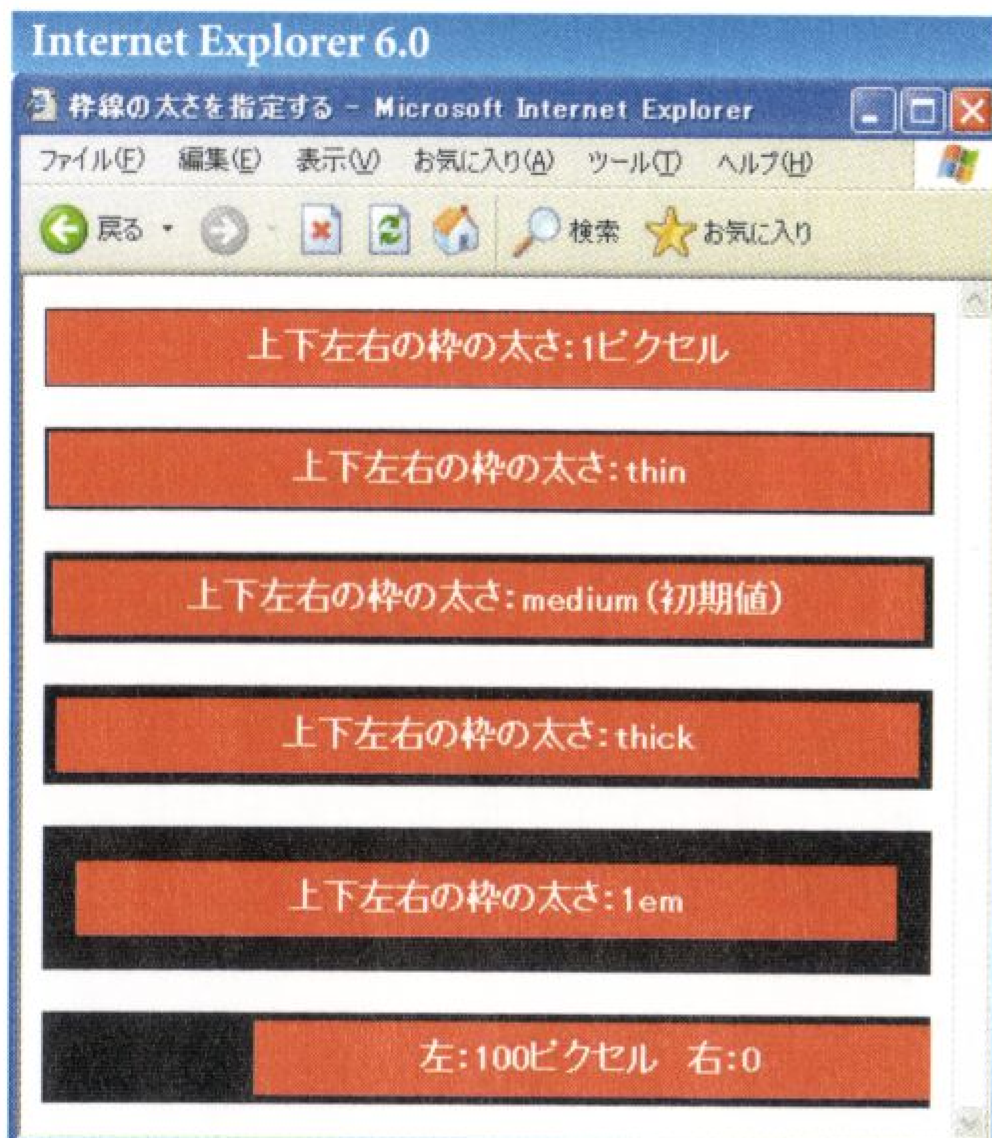
**【HTML】**

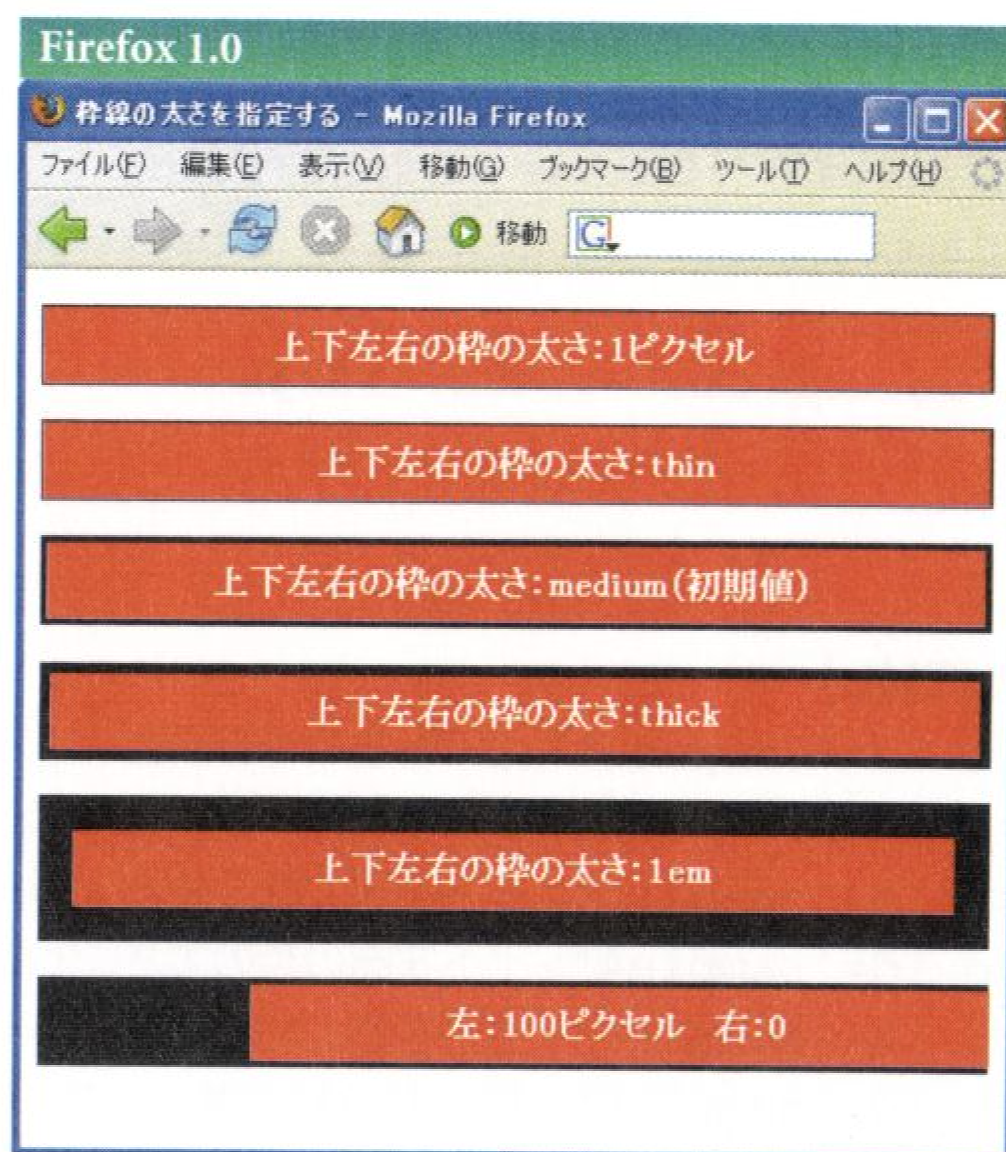
```
<p id="sample1">
この段落では、内容と枠線の間を上下左右とも0ピクセルに設定しています。
</p>
<p id="sample2">
この段落では、内容と枠線の間を上下左右とも20ピクセルに設定しています。
</p>
<p id="sample3">
この段落では、内容と枠線の間を上下左右とも40ピクセルに設定しています。
</p>
<p id="sample4">
この段落では、右側の内容と枠線の間を50%に設定しています。
</p>
<p id="sample5">
この段落では、左側の内容と枠線の間を50%に設定しています。
</p>
```

# 枠線の太さを指定する

border-top-width: 太さ ←上の枠線の太さ
border-bottom-width: 太さ ←下の枠線の太さ
border-left-width: 太さ ←左の枠線の太さ
border-right-width: 太さ ←右の枠線の太さ
border-width: 太さ ←上・右・下・左の枠線の太さ

| | |
|---|---|
| 太さ | 単位付きの数値・thin・medium・thick |




これらのプロパティは、枠線の太さを設定します。
「border-width」を利用すると、上下左右の枠線の太さを一度に設定することができます。その場合、値を半角スペースで区切って指定しますが、与えられた値の個数によって、次のように枠線の太さが設定されます。

- 値が１つの場合　値１→上下左右
- 値が２つの場合　値１→上下　値２→左右
- 値が３つの場合　値１→上　値２→左右　値３→下
- 値が４つの場合　値１→上　値２→右　値３→下　値４→左

「thin」「medium」「thick」は、それぞれ「細い枠線」「中くらいの枠線」「太い枠線」に設定します。この場合の実際の太さは、ブラウザによって異なります。

## Sample

**【CSS】**

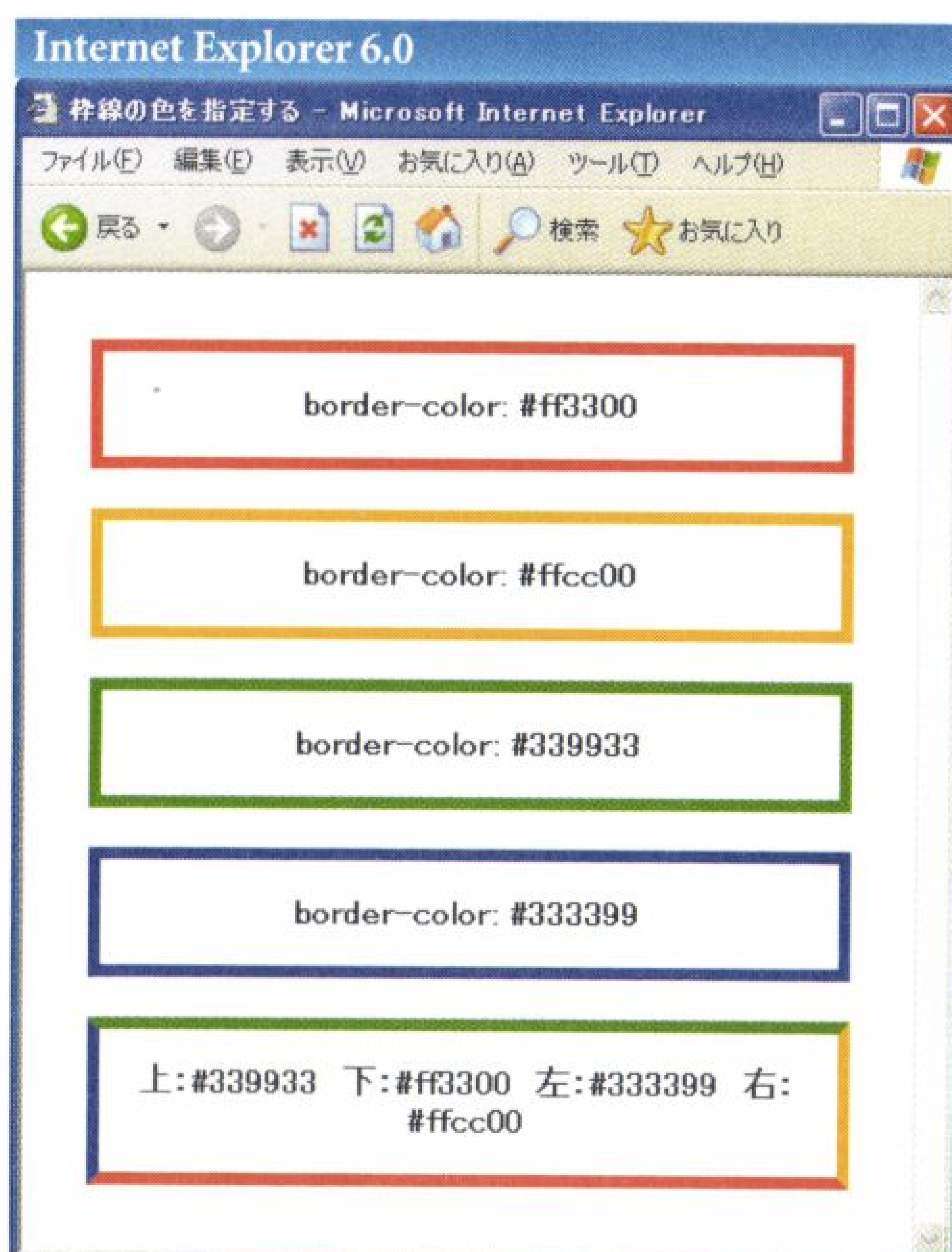
```
p {
  text-align: center;
  font-weight: bold;
  padding: 0.5em;
  border: solid #000000;
  color: #ffffff;
  background: #ff3300
}
#sample1 { border-width: 1px }
#sample2 { border-width: thin }
#sample3 { border-width: medium }
#sample4 { border-width: thick }
#sample5 { border-width: 1em }
#sample6 {
  border-left-width: 100px;
  border-right-width: 0;
}
```

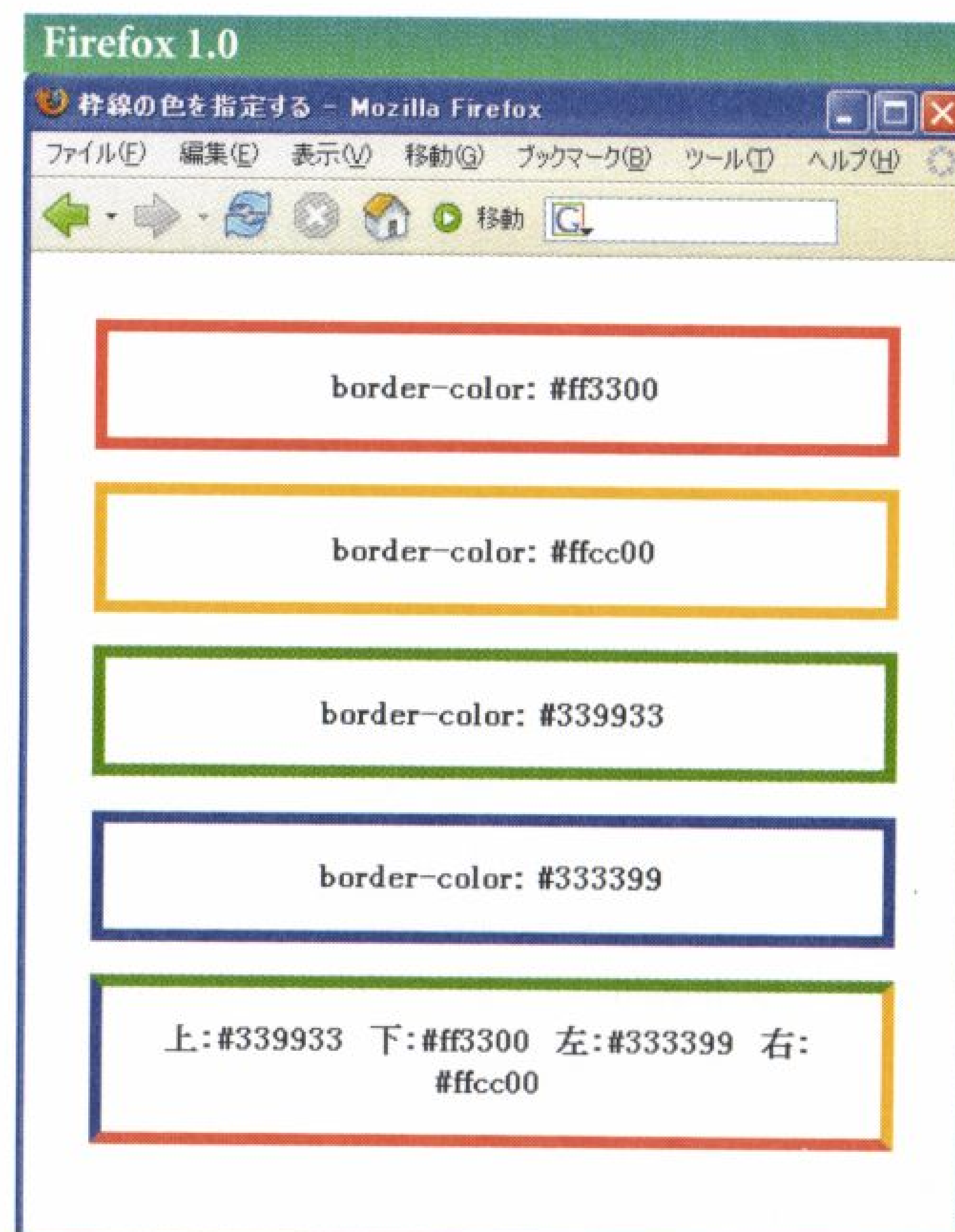
**【HTML】**

```
<p id="sample1">上下左右の枠の太さ：1ピクセル</p>
<p id="sample2">上下左右の枠の太さ：thin</p>
<p id="sample3">上下左右の枠の太さ：medium(初期値)</p>
<p id="sample4">上下左右の枠の太さ：thick</p>
<p id="sample5">上下左右の枠の太さ：1em</p>
<p id="sample6">左：100ピクセル　右：0</p>
```

# 枠線の色を指定する

**border-top-color:** *色指定* ◆上の枠線の色
**border-bottom-color:** *色指定* ◆下の枠線の色
**border-left-color:** *色指定* ◆左の枠線の色
**border-right-color:** *色指定* ◆右の枠線の色
**border-color:** *色指定* ◆上・右・下・左の枠線の色




これらのプロパティは、枠線の色を設定します。
「border-color」を利用すると、上下左右の枠線の色を一度に設定することができます。その場合、値を半角スペースで区切って指定しますが、与えられた値の個数によって、次のように枠線の色が設定されます。

| | |
|---|---|
| ・値が１つの場合 | 値１→上下左右 |
| ・値が２つの場合 | 値１→上下　値２→左右 |
| ・値が３つの場合 | 値１→上　値２→左右　値３→下 |
| ・値が４つの場合 | 値１→上　値２→右　値３→下　値４→左 |

なお、この値の初期値は、「color: *色指定*」で設定されている値となります。

## Sample

**【CSS】**

```
p {
  text-align: center;
  font-weight: bold;
  padding: 1em;
  border: solid 6px;
}
#sample1 { border-color: #ff3300 }
#sample2 { border-color: #ffcc00 }
#sample3 { border-color: #339933 }
#sample4 { border-color: #333399 }
#sample5 {
  border-top-color: #339933;
  border-bottom-color: #ff3300;
  border-left-color: #333399;
  border-right-color: #ffcc00
}
```

**【HTML】**

```
<p id="sample1">border-color: #ff3300</p>
<p id="sample2">border-color: #ffcc00</p>
<p id="sample3">border-color: #339933</p>
<p id="sample4">border-color: #333399</p>
<p id="sample5">上：#339933　下：#ff3300　左：#333399　右：#ffcc00</p>
```



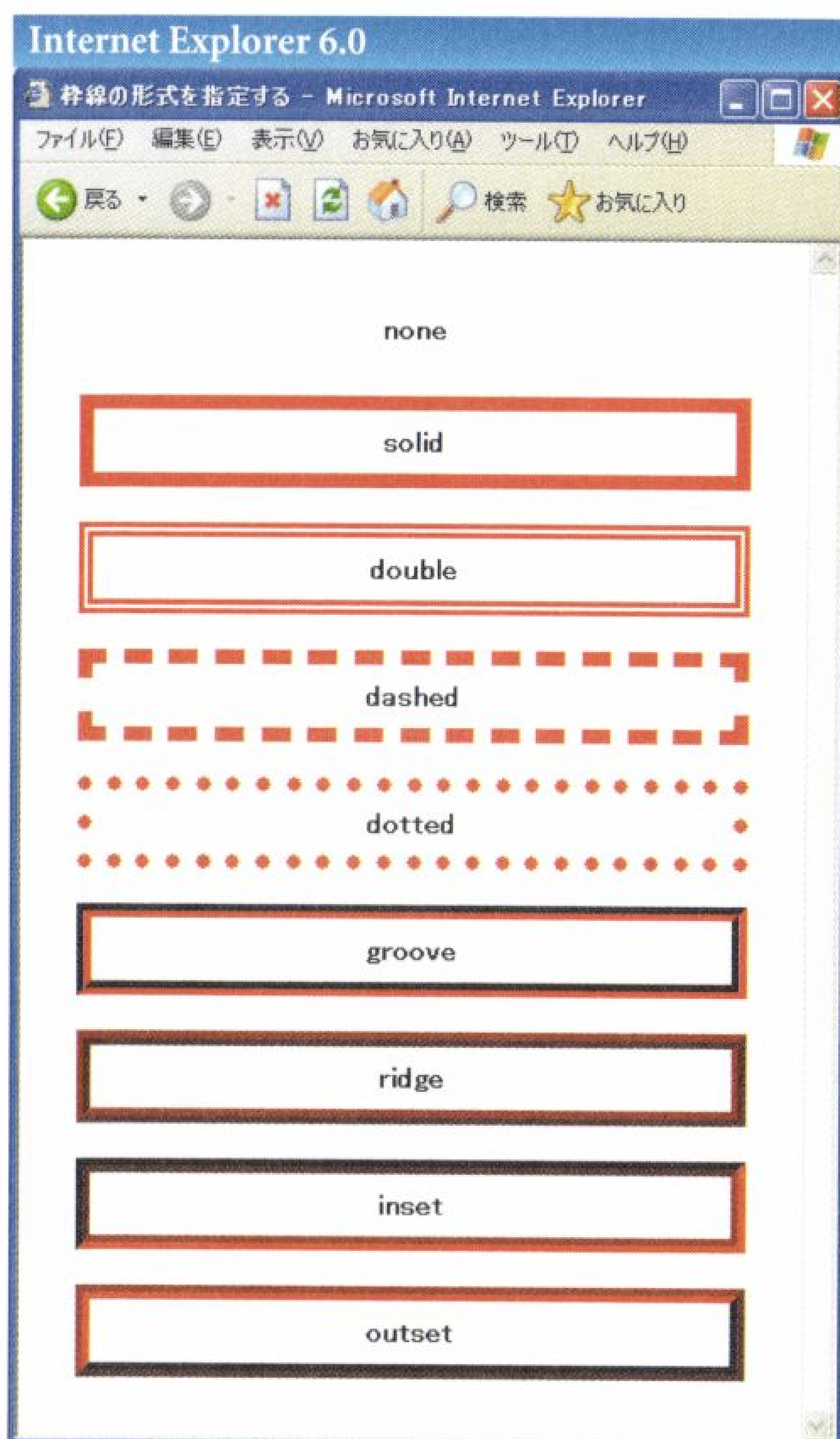
色指定：「CSSについて」の「色の指定方法」(P.180)
色指定：巻末付録「カラーチャート1～3」(巻末)

# 枠線の形式を指定する

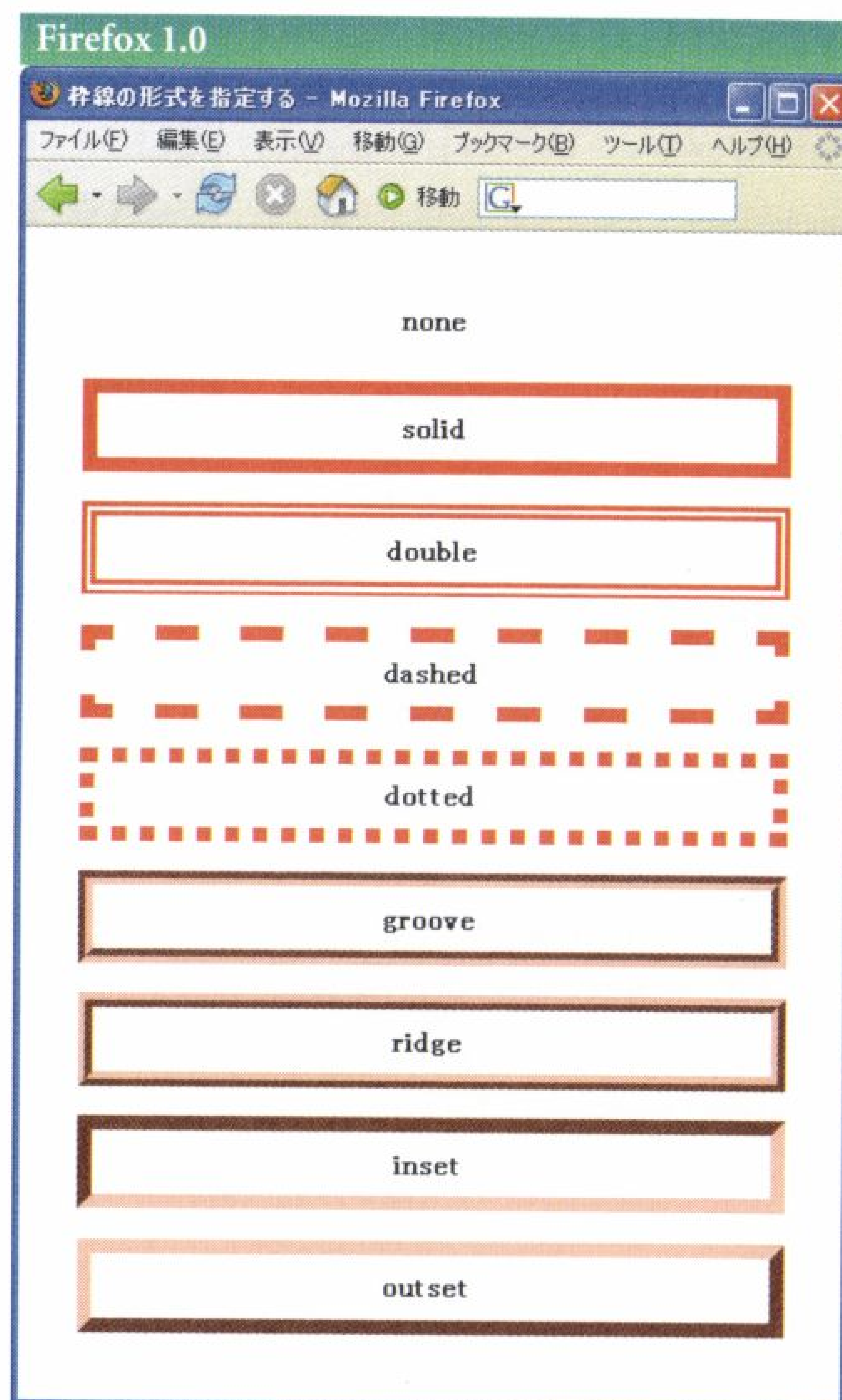
| | |
|---|---|
| **border-top-style:** *形式* | ⬅上の枠線の形式 |
| **border-bottom-style:** *形式* | ⬅下の枠線の形式 |
| **border-left-style:** *形式* | ⬅左の枠線の形式 |
| **border-right-style:** *形式* | ⬅右の枠線の形式 |
| **border-style:** *形式* | ⬅上・右・下・左の枠線の形式 |

| | |
|---|---|
| 形式 | none・hidden・dotted・dashed・solid・double・groove・ridge・inset・outset |

Internet Explorer 6.0




Firefox 1.0




これらのプロパティは、枠線の形式を設定します。
「border-style」を利用すると、上下左右の枠線の形式を一度に設定することができます。その場合、値を半角スペースで区切って指定しますが、与えられた値の個数によって、次のように枠線の形式が設定されます。

| | |
|---|---|
| ・値が１つの場合 | 値１→上下左右 |
| ・値が２つの場合 | 値１→上下　値２→左右 |
| ・値が３つの場合 | 値１→上　値２→左右　値３→下 |
| ・値が４つの場合 | 値１→上　値２→右　値３→下　値４→左 |

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

「none」と「hidden」はどちらも枠線を表示せず、枠線の太さも0に設定します。ただし、テーブルのセルの枠線として重なりあった場合には、「none」は他の値を優先し、「hidden」は自分自身の値を優先します。この値の初期値は「none」です。

## Sample

**【CSS】**

```
p {
  text-align: center;
  font-weight: bold;
  padding: 0.5em;
  border: solid 8px #ff3300;
}
#sample1 { border-style: none }
#sample2 { border-style: solid }
#sample3 { border-style: double }
#sample4 { border-style: dashed }
#sample5 { border-style: dotted }
#sample6 { border-style: groove }
#sample7 { border-style: ridge }
#sample8 { border-style: inset }
#sample9 { border-style: outset }
```

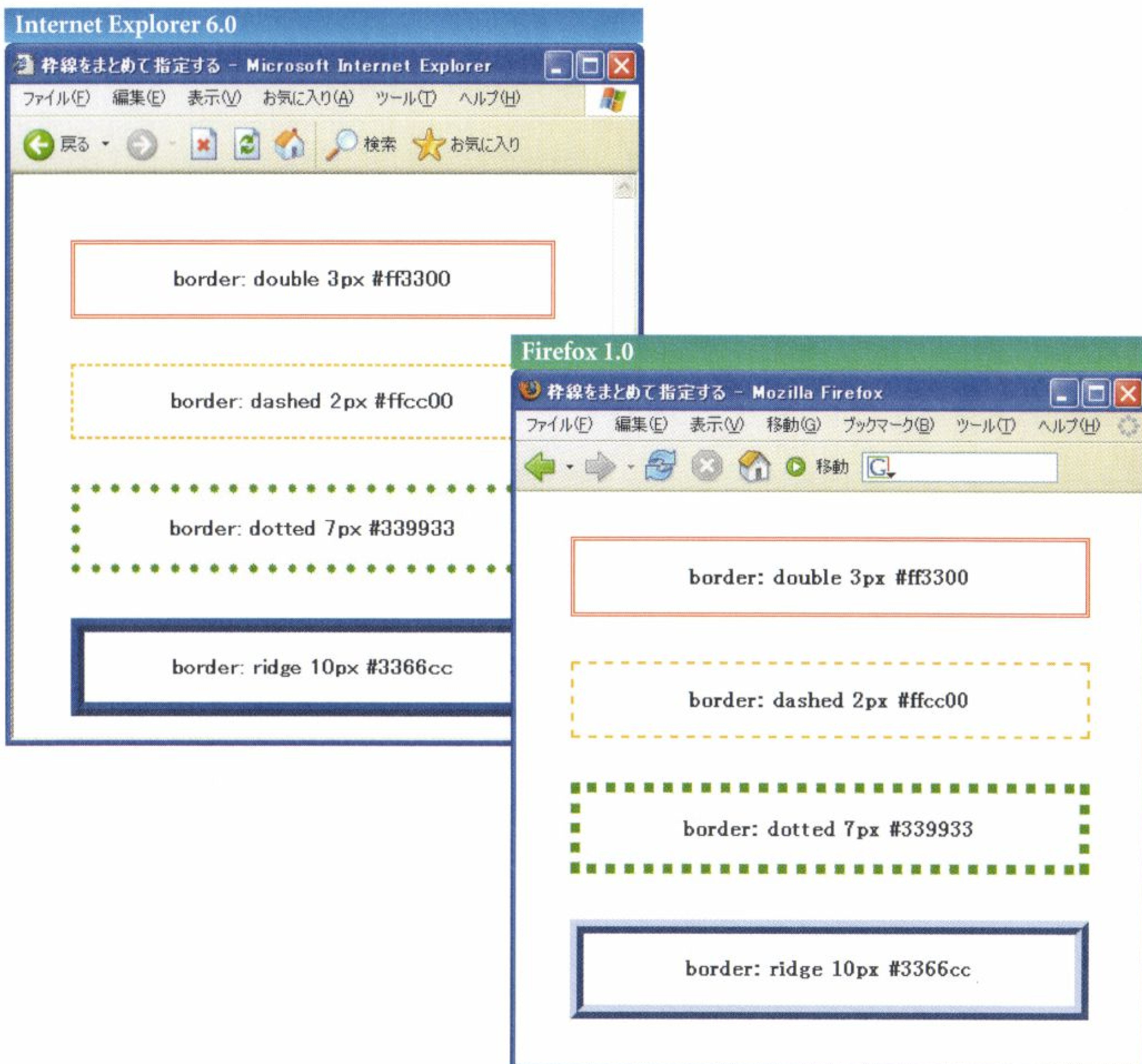
**【HTML】**

```
<p id="sample1">none</p>
<p id="sample2">solid</p>
<p id="sample3">double</p>
<p id="sample4">dashed</p>
<p id="sample5">dotted</p>
<p id="sample6">groove</p>
<p id="sample7">ridge</p>
<p id="sample8">inset</p>
<p id="sample9">outset</p>
```

# 枠線をまとめて指定する

**border-top:** *枠線関連のプロパティの値* ←上の枠線の設定
**border-bottom:** *枠線関連のプロパティの値* ←下の枠線の設定
**border-left:** *枠線関連のプロパティの値* ←左の枠線の設定
**border-right:** *枠線関連のプロパティの値* ←右の枠線の設定
**border:** *枠線関連のプロパティの値* ←上下左右の枠線に同じ値を設定

| 【枠線関連のプロパティの値】 |
| --- |
| border-color(P.243)で指定できる値 |
| border-width(P.241)で指定できる値 |
| border-style(P.245)で指定できる値 |




これらのプロパティは、枠線関連のプロパティの値をまとめて指定します。
必要な値を任意の順序で半角スペースで区切って指定します。指定しなかった値については、初期値が指定されたことになります。
なお、「border」を使用して上下左右に別々の設定をすることはできません。

## Sample

**【CSS】**

```
p {
  text-align: center;
  font-weight: bold;
  margin: 2em;
  padding: 1em
}
#sample1 { border: double  3px #ff3300 }
#sample2 { border: dashed  2px #ffcc00 }
#sample3 { border: dotted  7px #339933 }
#sample4 { border: ridge  10px #3366cc }
```

**【HTML】**

```
<p id="sample1">border: double  3px #ff3300</p>
<p id="sample2">border: dashed  2px #ffcc00</p>
<p id="sample3">border: dotted  7px #339933</p>
<p id="sample4">border: ridge  10px #3366cc</p>
```

# 幅と高さを指定する

**width:** *幅*
**height:** *高さ*

幅・高さ　単位付きの数値・%・auto

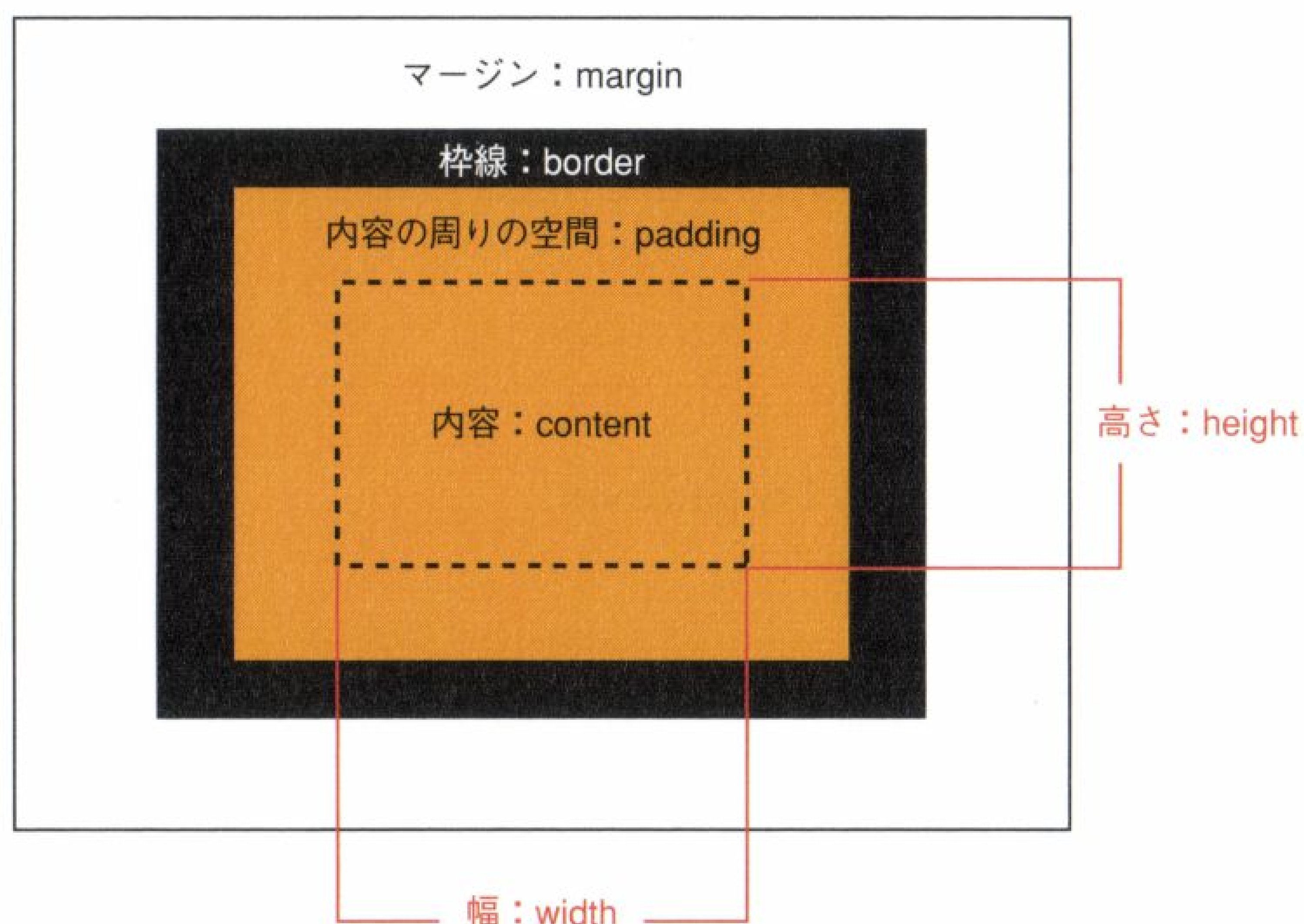
Internet Explorer 6.0




Firefox 1.0




**ボックスの構造**



width プロパティと height プロパティは、内容を表示する部分の幅と高さを設定します。ブロックレベル要素と置換要素(img や input、textarea、select など)の他、横列と横列グループ(tr・thead・tbody・tfoot)を除くテーブル関連要素に対して指定できます。% で指定する場合は、そのボックスを含むボックスの幅または高さに対する割合になります。値として「auto」を指定すると状況に応じて自動的に調整されますが、置換要素の場合には本来の幅や高さになります。
なお、Windows 版の Internet Explorer 5.5 までと Internet Explorer 6.0 の互換モードでは、内容の周りの空間と枠線も含んだ範囲に対して幅や高さが設定されてしまいますので、注意してください。

## Sample

【CSS】

```
img.small  { width:  50px; height:  50px }
img.normal { width:  auto; height:  auto }
img.large  { width: 200px; height: 200px }
.half {
  width: 50%;
  color: #ffffff;
  background: #ff3300
}
```

【HTML】

```
<p>
※中央がオリジナルサイズ(100px×100px)
</p>
<p>
<img src="orange.jpg" alt="" class="small">
<img src="orange.jpg" alt="" class="normal">
<img src="orange.jpg" alt="" class="large">
</p>

<hr>

<p class="half">
これ以下のサンプルでは、要素の幅(width)を「50%」に設定しています。
</p>
<p>
select 要素：<br>
<select class="half">
<option selected>選択項目1</option>
<option>選択項目2</option>
</select>
</p>
<p>
input 要素：<br>
<input type="text" class="half" value="入力フィールド">
</p>
```

```
<p>
textarea要素：<br>
<textarea rows="5" cols="30" class="half">
複数行の入力フィールド
</textarea>
</p>
```

HTML：TIPS「新しいブラウザは<!DOCTYPE>の書き方で表示が変わる！」(P.9)

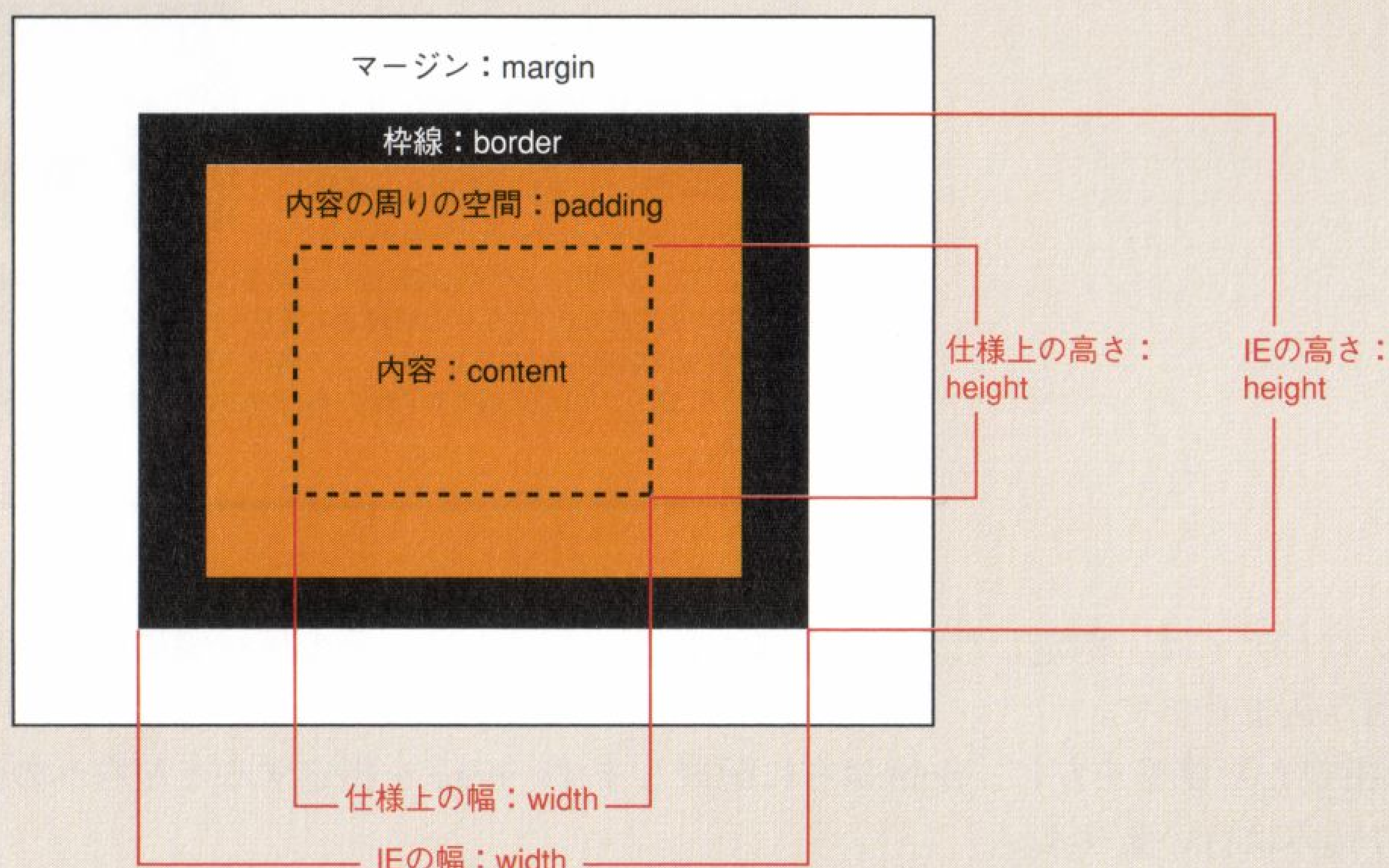


## コラム

### Internet Explorerの幅と高さの適用範囲に注意！

左ページの解説にも書きましたが、Windows版のInternet Explorer 4.0～5.5と6.0の互換モードでは、widthプロパティとheightプロパティの適用される範囲が標準的な仕様とは異なっています。具体的には、本来はボックスの内容部分の幅や高さを設定するはずのものが、Internet Explorerでは枠線と内容の周りの空間も含んだ領域の幅や高さとして設定されます。特にボックスの枠線を太くしたり内容の周りの空間を広くとっている場合には、正しい表示をするブラウザとは全然違う表示結果となりますので、注意してください。

**ボックスの構造**



IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 左右への配置と回り込みを指定する

**float: 配置位置**

配置位置　　left · right · none




floatプロパティは、指定した要素を左または右に配置して、その反対側に後に続く要素を回り込ませます。
leftは指定した要素を左に、rightは右に配置します。noneを指定すると左右への配置と回り込みは行いません。
回り込みを指定した後にそれを解除するためには、clearプロパティを利用します。

表示と配置の指定をする

## Sample

**【CSS】**

```
img.left {
  float: left;
  margin-right: 0.8em;
  margin-bottom: 0.5em
}
img.right {
  float: right;
  margin-left: 0.8em;
  margin-bottom: 0.5em
}
p {
  clear: both;
  line-height: 1.4
}
```

**【HTML】**

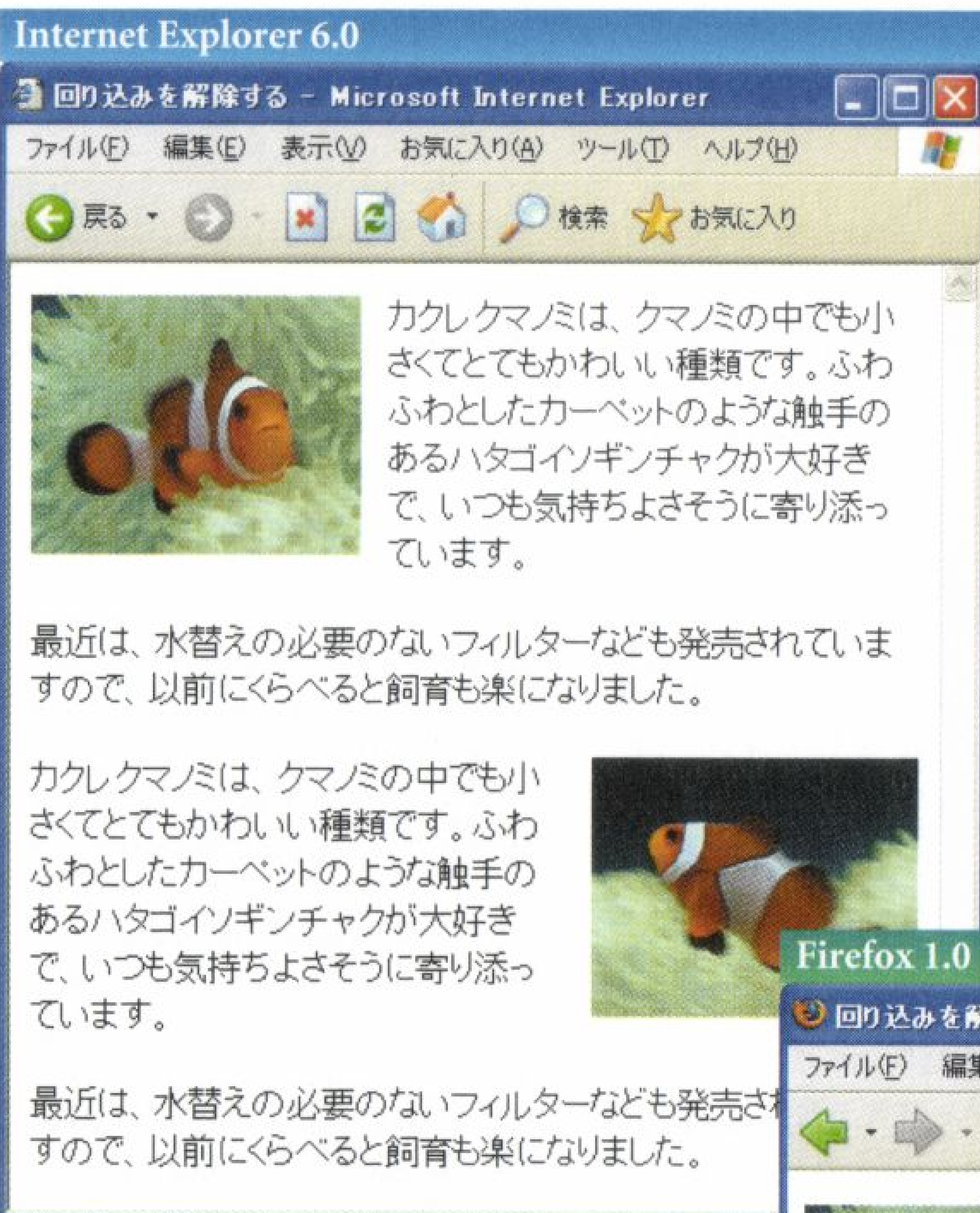
```
<p>
<img src="fish1.jpg" class="left" width="155" height="120" alt="">
カクレクマノミは、クマノミの中でも小さくてとてもかわいい種類です。ふわふわとしたカーペットのような触手のあるハタゴイソギンチャクが大好きで、いつも気持ちよさそうに寄り添っています。最近は、水替えの必要のないフィルターなども発売されていますので、以前にくらべると飼育も楽になりました。また、様々な付着生物が生きたまま付いている珊瑚礁の岩を入れておくことで、さらに水槽内の環境が良くなるようです。この岩は、ライブロックと呼ばれており、現在では主に海外から輸入されたものが販売されているようです。
</p>
<p>
<img src="fish2.jpg" class="right" width="155" height="120" alt="">
カクレクマノミは、クマノミの中でも小さくてとてもかわいい種類です。ふわふわとしたカーペットのような触手のあるハタゴイソギンチャクが大好きで、いつも気持ちよさそうに寄り添っています。最近は、水替えの必要のないフィルターなども発売されていますので、以前にくらべると飼育も楽になりました。また、様々な付着生物が生きたまま付いている珊瑚礁の岩を入れておくことで、さらに水槽内の環境が良くなるようです。この岩は、ライブロックと呼ばれており、現在では主に海外から輸入されたものが販売されているようです。
</p>
```

# 回り込みを解除する

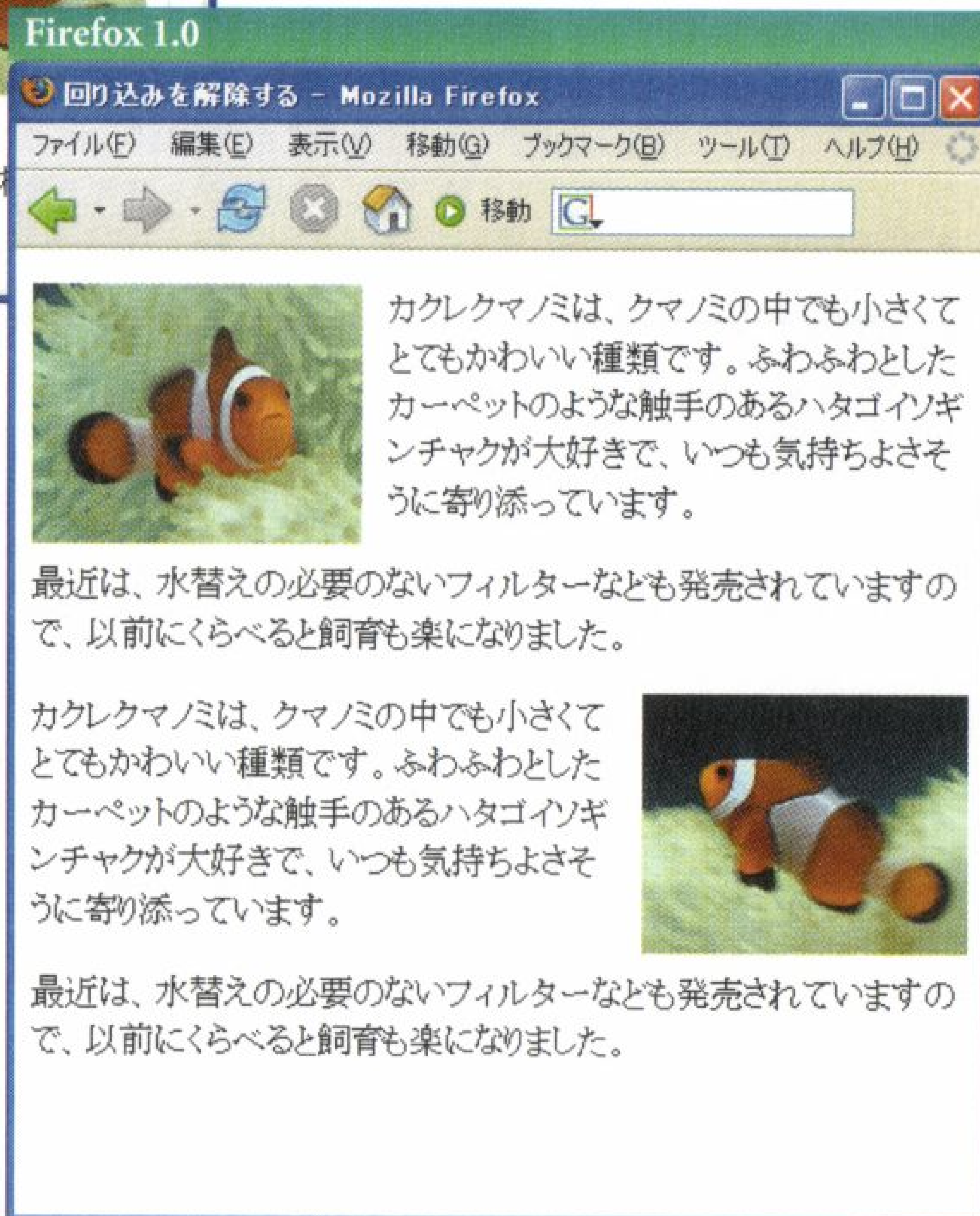
## clear: どちら側の要素に対して解除するか

| 【どちら側の要素に対して解除するか】 | |
|---|---|
| left | 左側の要素に対する回り込みを解除 |
| right | 右側の要素に対する回り込みを解除 |
| both | 両側の要素に対する回り込みを解除 |
| none | 回り込みを解除しない |

Internet Explorer 6.0




Firefox 1.0



clear プロパティは、ある要素を左または右に配置してテキストなどを回り込ませた場合の、回り込みを解除します。
ブロックレベルの要素に対して指定することができます。

## Sample

**【CSS】**

```
img.left {
  float: left;
  margin-right: 0.8em;
  margin-bottom: 0.5em
}
img.right {
  float: right;
  margin-left: 0.8em;
  margin-bottom: 0.5em
}
p {
  clear: both;
  line-height: 1.4
}
```

**【HTML】**

```
<p>
<img src="fish1.jpg" class="left" width="155" height="120" alt="">
カクレクマノミは、クマノミの中でも小さくてとてもかわいい種類です。ふわふわとしたカーペット
のような触手のあるハタゴイソギンチャクが大好きで、いつも気持ちよさそうに寄り添っています。
</p>
<p>
最近は、水替えの必要のないフィルターなども発売されていますので、以前にくらべると飼育も楽
になりました。
</p>
<p>
<img src="fish2.jpg" class="right" width="155" height="120" alt="">
カクレクマノミは、クマノミの中でも小さくてとてもかわいい種類です。ふわふわとしたカーペット
のような触手のあるハタゴイソギンチャクが大好きで、いつも気持ちよさそうに寄り添っています。
</p>
<p>
最近は、水替えの必要のないフィルターなども発売されていますので、以前にくらべると飼育も楽
になりました。
</p>
```

# センタリングする

**margin-left: auto; margin-right: auto**
**margin: auto**
**margin:** *上下マージン* **auto**
**margin:** *上マージン* **auto** *下マージン*

※上下マージン、上マージン、下マージンについては任意の値

Internet Explorer 6.0




Firefox 1.0




これらの用法は、ブロックレベル要素のボックスをセンタリングさせる指定です。左右のマージンの値を「auto」に設定するとセンタリングされますので、上の書式のどれを使用してもセンタリングは可能です。ブロックレベル要素の内容の行揃えを設定したい場合には、text-alignプロパティを使用してください。

なお、Windows版のInternet Explorer 6.0の場合は、標準モードでなければセンタリングされませんので、注意してください。

## Sample

**【CSS】**

```
h1, p {
  margin-left: auto;
  margin-right: auto;
  padding: 20px;
  width: 150px;
}
table {
  margin: auto
}
h1, p, table {
  color: #ffffff;
  background: #ff6600
}
```

**【HTML】**

```
<h1>これはh1要素です</h1>

<p>これはp要素です。</p>

<table border="3" cellpadding="6">
<tr><th>ヘッダ</th><th>ヘッダ</th><th>ヘッダ</th></tr>
<tr><td>データ</td><td>データ</td><td>データ</td></tr>
</table>
```


HTML：TIPS「新しいブラウザは<!DOCTYPE>の書き方で表示が変わる！」(P.9)
text-align：「テキスト」の「行揃えを指定する」(P.214)
margin-left・margin-right・margin：「ボックス」の「マージンを設定する」(P.237)

CSS

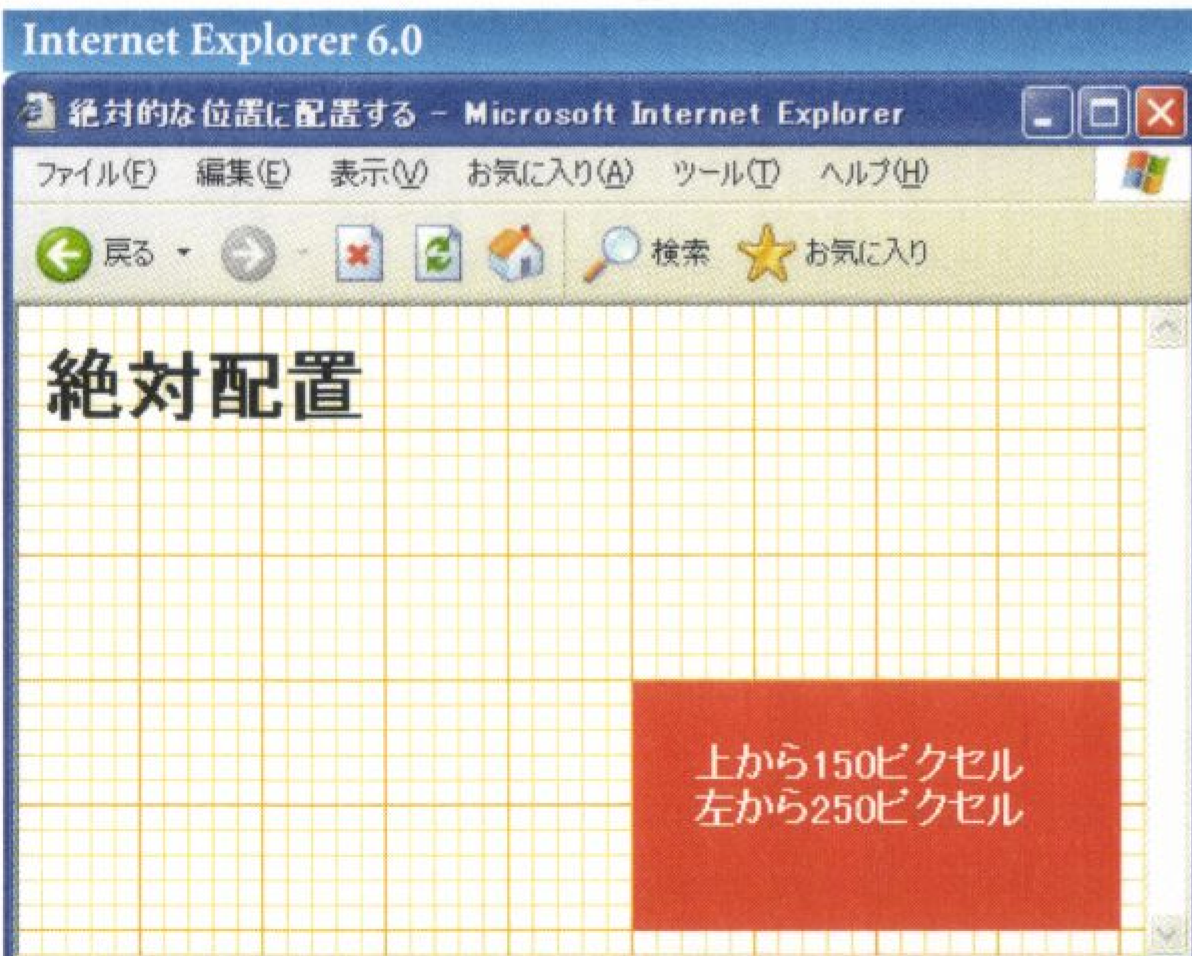
IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 絶対的な位置に配置する

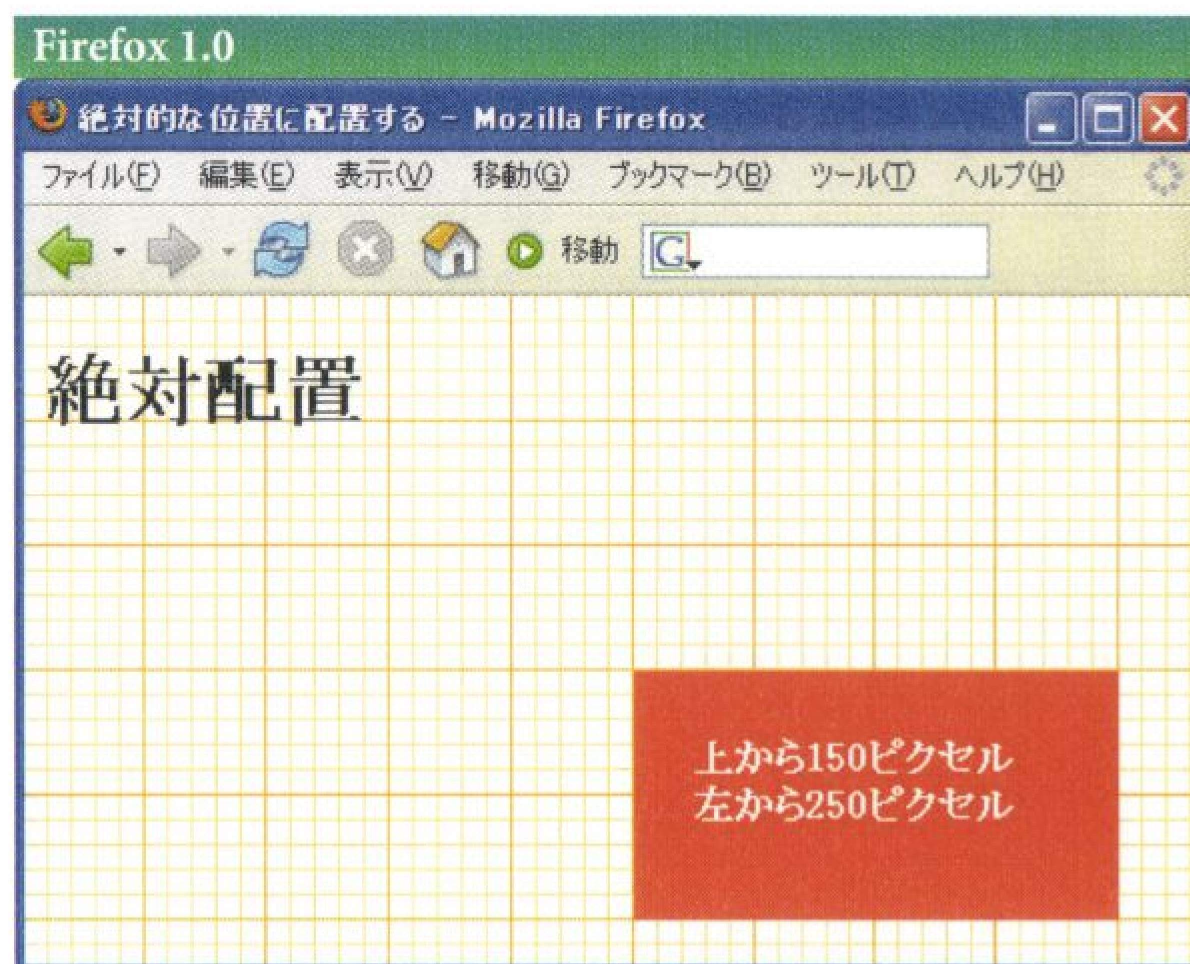
| | |
|---|---|
| **position: absolute** | ←絶対配置であることを示す |
| **top:** *距離* | ←上からの距離 |
| **bottom:** *距離* | ←下からの距離 |
| **left:** *距離* | ←左からの距離 |
| **right:** *距離* | ←右からの距離 |

| | |
|---|---|
| 距離 | 単位付きの数値・% |

Internet Explorer 6.0




Firefox 1.0




「position: absolute」と距離を指定するプロパティは、指定された要素を絶対的な位置で指定して配置します。
この指定をすると、その部分は通常の配置とは別に扱われるようになるため、他の要素の配置には一切影響を与えなくなります。top プロパティは親ボックスの上から指定した要素のボックスの上までの距離、bottom プロパティは親ボックスの下から指定し

た要素のボックスの下までの距離、leftプロパティは親ボックスの左から指定した要素のボックスの左までの距離、rightプロパティは親ボックスの右から指定した要素のボックスの右までの距離を指定します。

## Sample

**【CSS】**

```
body {
  color: #000000;
  background: #ffffff url(grid.gif)
}
p {
  position: absolute;
  top: 150px;
  left: 250px;
  width: 150px;
  height: 50px;
  margin: 0;
  padding: 25px;
  font-weight: bold;
  color: #ffffff;
  background: #ff3300
}
```

**【HTML】**

```
<h1>絶対配置</h1>

<p>
上から150ピクセル<br>
左から250ピクセル
</p>
```

# 相対的な位置に配置する

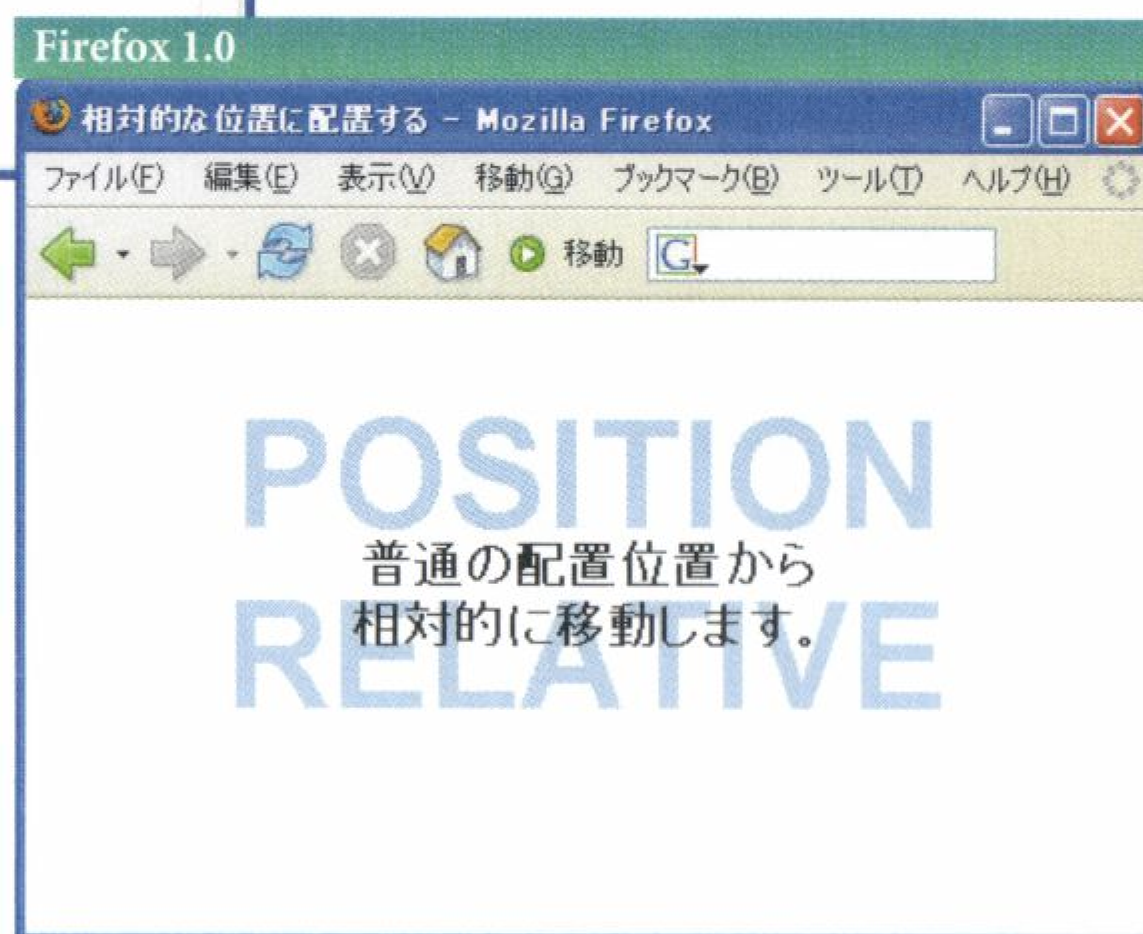
| | |
|---|---|
| **position: relative** | ←相対配置であることを示す |
| **top:** *距離* | ←上からの距離 |
| **bottom:** *距離* | ←下からの距離 |
| **left:** *距離* | ←左からの距離 |
| **right:** *距離* | ←右からの距離 |

| | |
|---|---|
| 距離 | 単位付きの数値・% |

Internet Explorer 6.0




Firefox 1.0




「position: relative」と距離を指定するプロパティは、指定された要素を本来表示されるべき位置から相対的に移動させて配置します。
この後に続く要素は、指定された要素が本来の位置に表示されている場合と同様に配置されます。top プロパティは上から下へ移動させる距離、bottom プロパティは下から上へ移動させる距離、left プロパティは左から右へ移動させる距離、right プロパティは右から左へ移動させる距離を指定します。

Sample

【CSS】

```
h1 {
  font: bold 60px Arial, sans-serif;
  text-align: center;
  margin-bottom: 0;
  color: #99ccff;
  background-color: #ffffff
}
p {
  position: relative;
  top: -90px;
  font: bold 20px "ＭＳ Ｐゴシック",Osaka,sans-serif;
  text-align: center;
  margin-top: 0;
  color: #000000;
  background-color: transparent
}
```

【HTML】

```
<h1>POSITION<br>RELATIVE</h1>

<p>普通の配置位置から<br>相対的に移動します。</p>
```

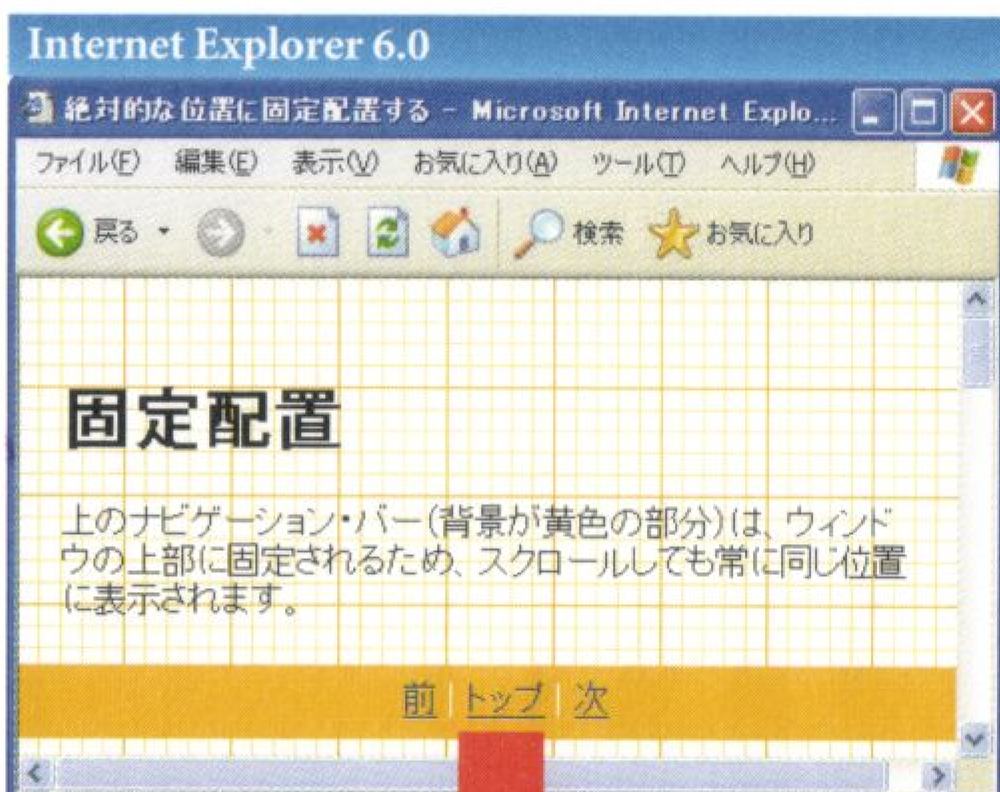
IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

# 絶対的な位置に固定配置する

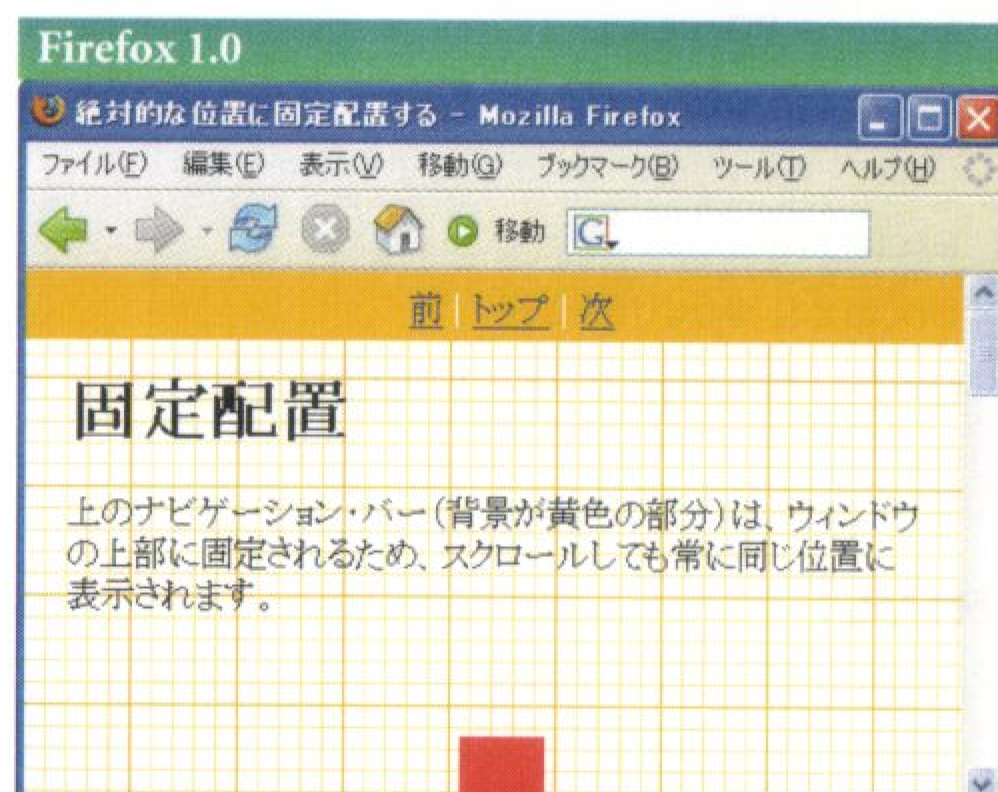
| | |
|---|---|
| **position: fixed** | ←固定配置であることを示す |
| **top:** *距離* | ←上からの距離 |
| **bottom:** *距離* | ←下からの距離 |
| **left:** *距離* | ←左からの距離 |
| **right:** *距離* | ←右からの距離 |

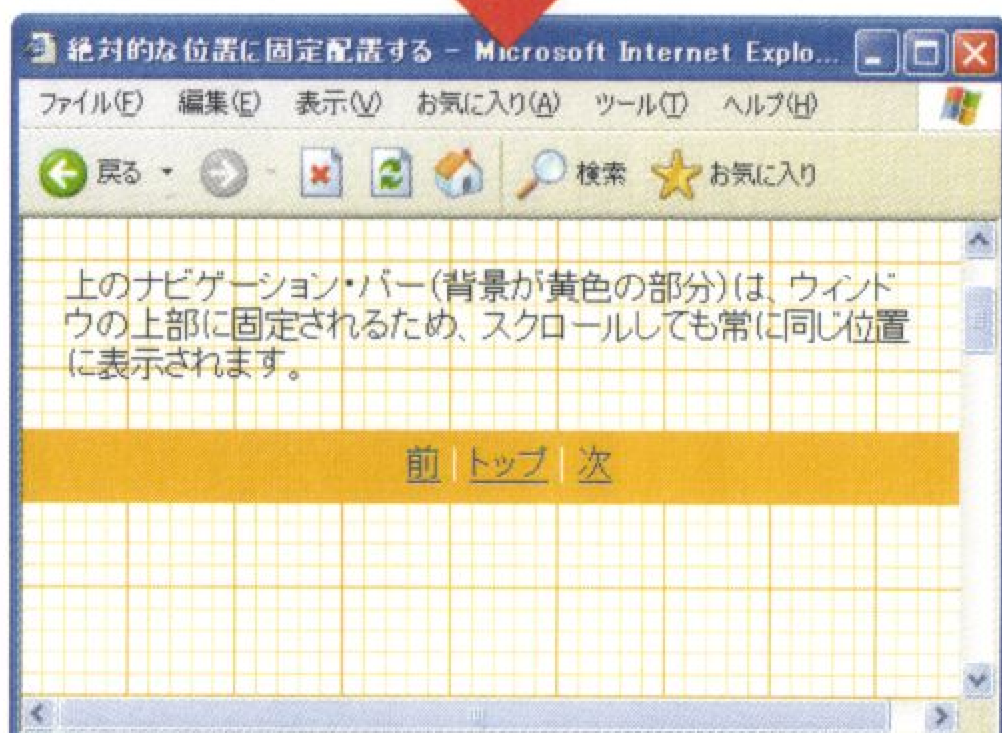
| | |
|---|---|
| 距離 | 単位付きの数値・% |

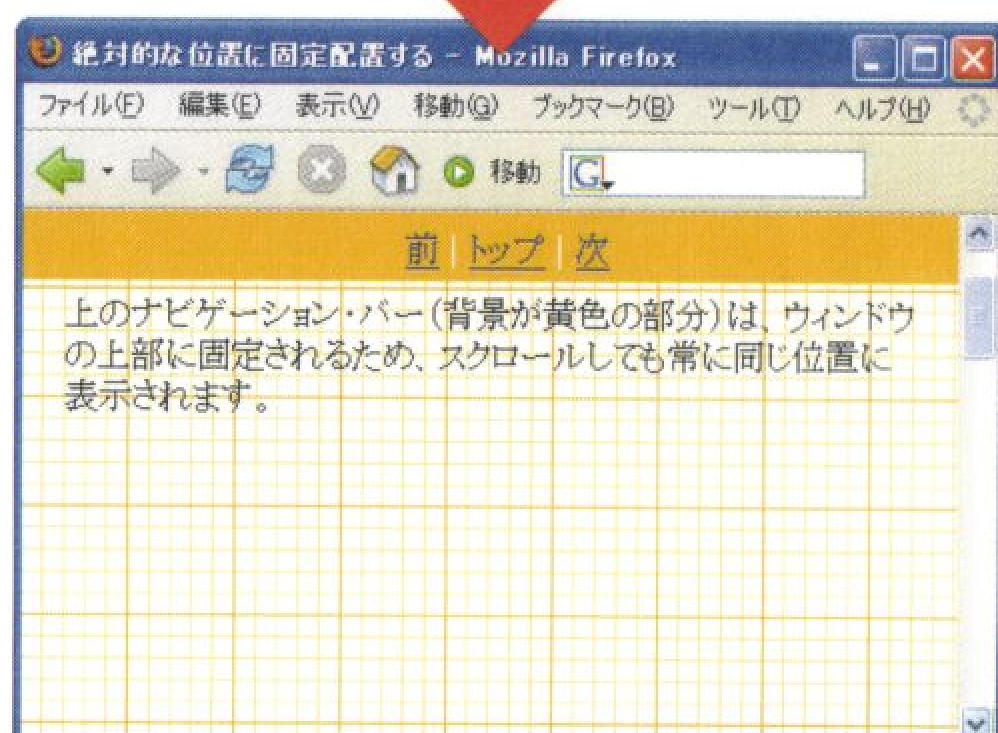
Internet Explorer 6.0




Firefox 1.0




「position: fixed」と距離を指定するプロパティは、指定された要素を絶対的な位置に配置しますが、その要素はウィンドウ上のその位置に固定されて、スクロールしても位置が変わらなくなります。
この指定をされた要素は、通常の配置とは別に扱われるようになるため、他の要素の配置には一切影響を与えません。top プロパティは親ボックスの上から指定した要素のボックスの上までの距離、bottom プロパティは親ボックスの下から指定した要素のボックスの下までの距離、left プロパティは親ボックスの左から指定した要素のボックスの左までの距離、right プロパティは親ボックスの右から指定した要素のボックスの右までの距離を指定します。

Sample

【CSS】

```
body {
  margin: 3em 0 1em;
  color: #000000;
  background: #ffffff url(grid.gif)
}
#navi {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 1em;
  margin: 0;
  padding: 0.5em;
  text-align: center;
  color: #ffffff;
  background: #ffcc00;
}
h1, p {
  margin-left: 20px;
  margin-right: 20px;
}
```

【HTML】

```
<h1>固定配置</h1>

<p>
上のナビゲーション・バー(背景が黄色の部分)は、ウィンドウの上部に固定されるため、スクロールしても常に同じ位置に表示されます。
</p>

<div id="navi">
<a href="prev.html">前</a> |
<a href="home.html">トップ</a> |
<a href="next.html">次</a>
</div>
```

Firefox
Mozilla
N7.X
N6.X
Opera7
Opera6
Safari
IE5-mac

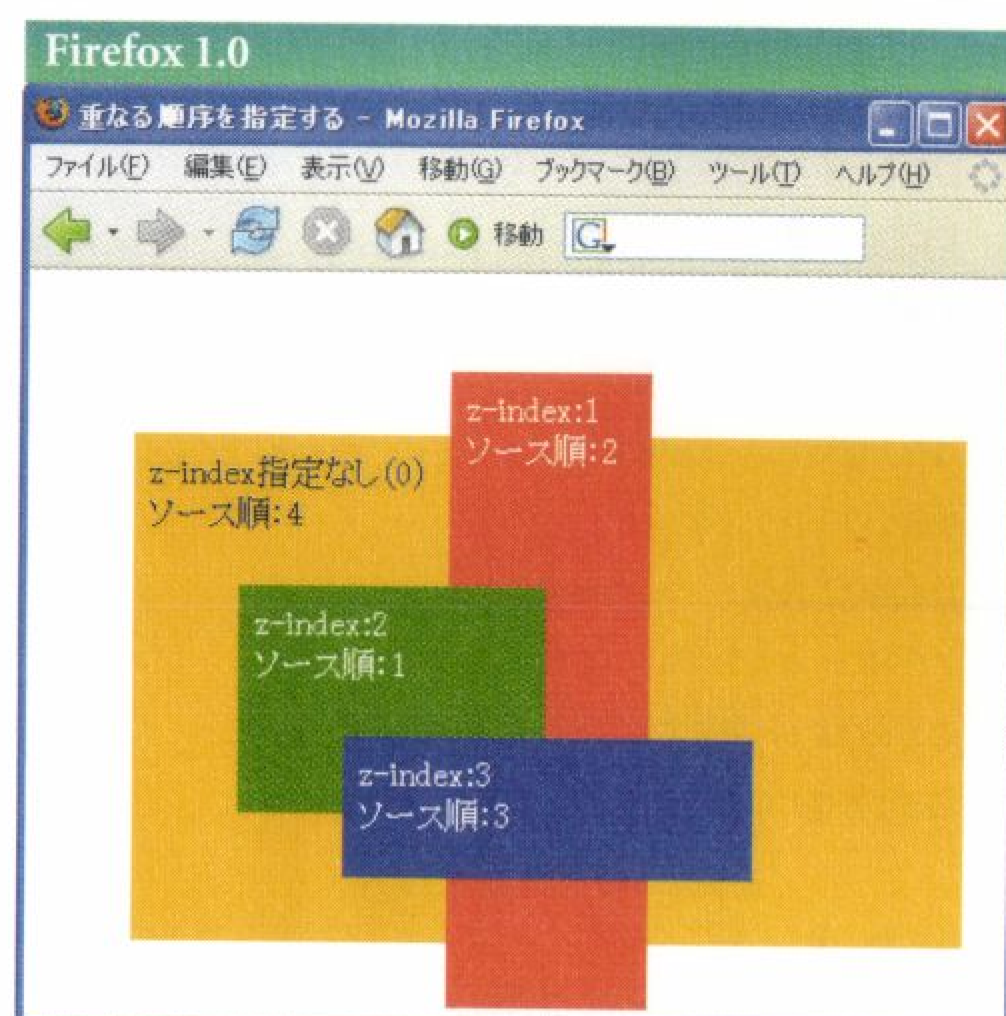
表示と配置の指定をする

# 重なる順序を指定する

## z-index: 重なる順序

重なる順序　整数値




z-index プロパティは、絶対配置や相対配置されている要素の重なる順序を指定します。通常表示される状態を0として、値が大きいものほど上に(重ねられた状態で)表示されます。

### Sample

【CSS】

```
#sample1 {
  position: absolute;
  z-index: 2;
  top: 130px;
  left: 100px;
  width: 130px;
  height: 90px;
  color: #ffffff;
  background-color: #339933
}
#sample2 {
  position: absolute;
  z-index: 1;
  top: 30px;
  left: 200px;
  width: 80px;
  height: 280px;
  color: #ffffff;
  background-color: #ff3300
```

```
}
#sample3 {
  position: absolute;
  z-index: 3;
  top: 200px;
  left: 150px;
  width: 180px;
  height: 50px;
  color: #ffffff;
  background-color: #333399
}
#sample4 {
  position: absolute;
  top: 60px;
  left: 50px;
  width: 380px;
  height: 220px;
  color: #000000;
  background-color: #ffcc00
}
p { padding: 0.5em }
```

**【HTML】**

```
<p id="sample1">
z-index:2<br>ソース順：1
</p>

<p id="sample2">
z-index:1<br>ソース順：2
</p>

<p id="sample3">
z-index:3<br>ソース順：3
</p>

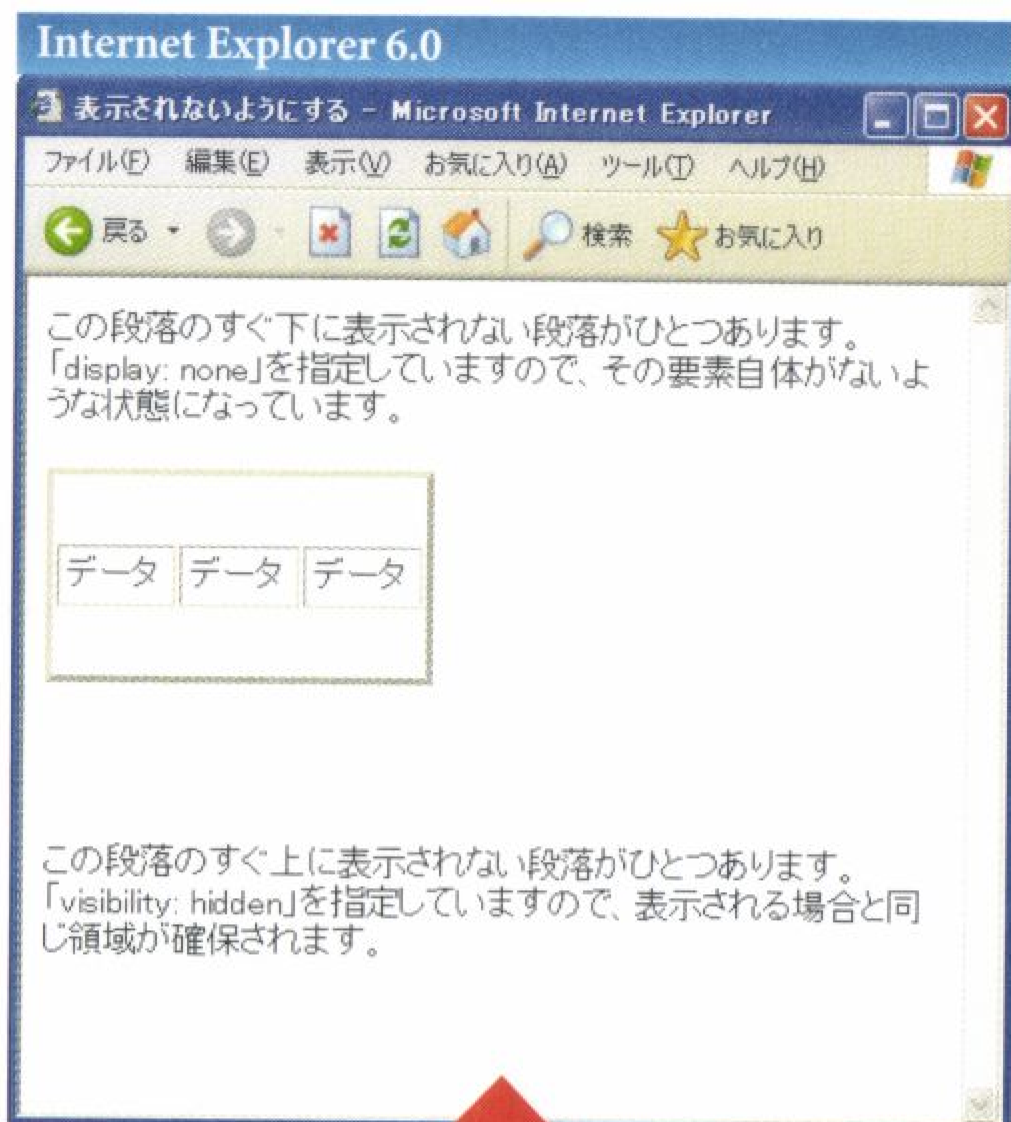
<p id="sample4">
z-index指定なし(0)<br>ソース順：4
</p>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.X Opera7 Opera6 Safari IE5-mac

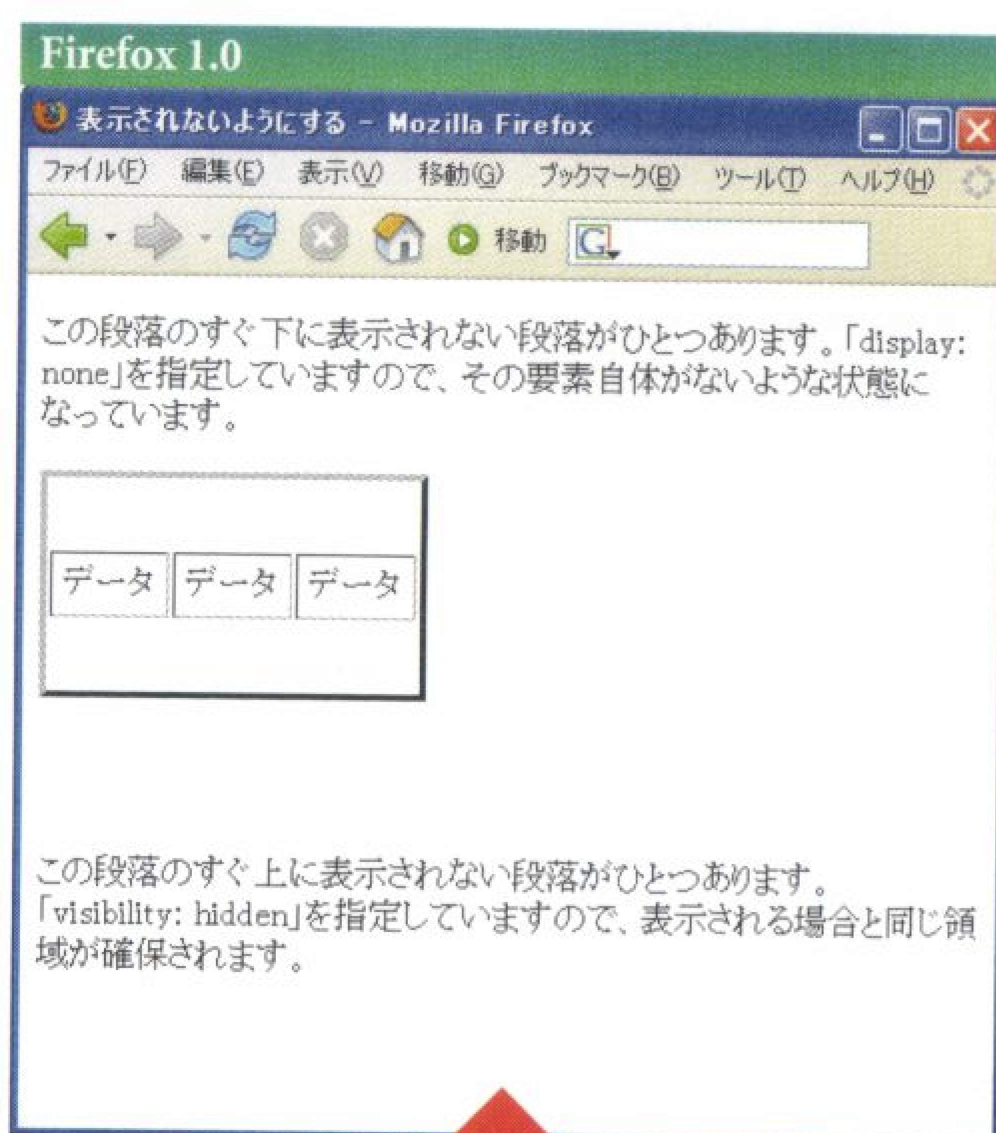
# 表示されないようにする

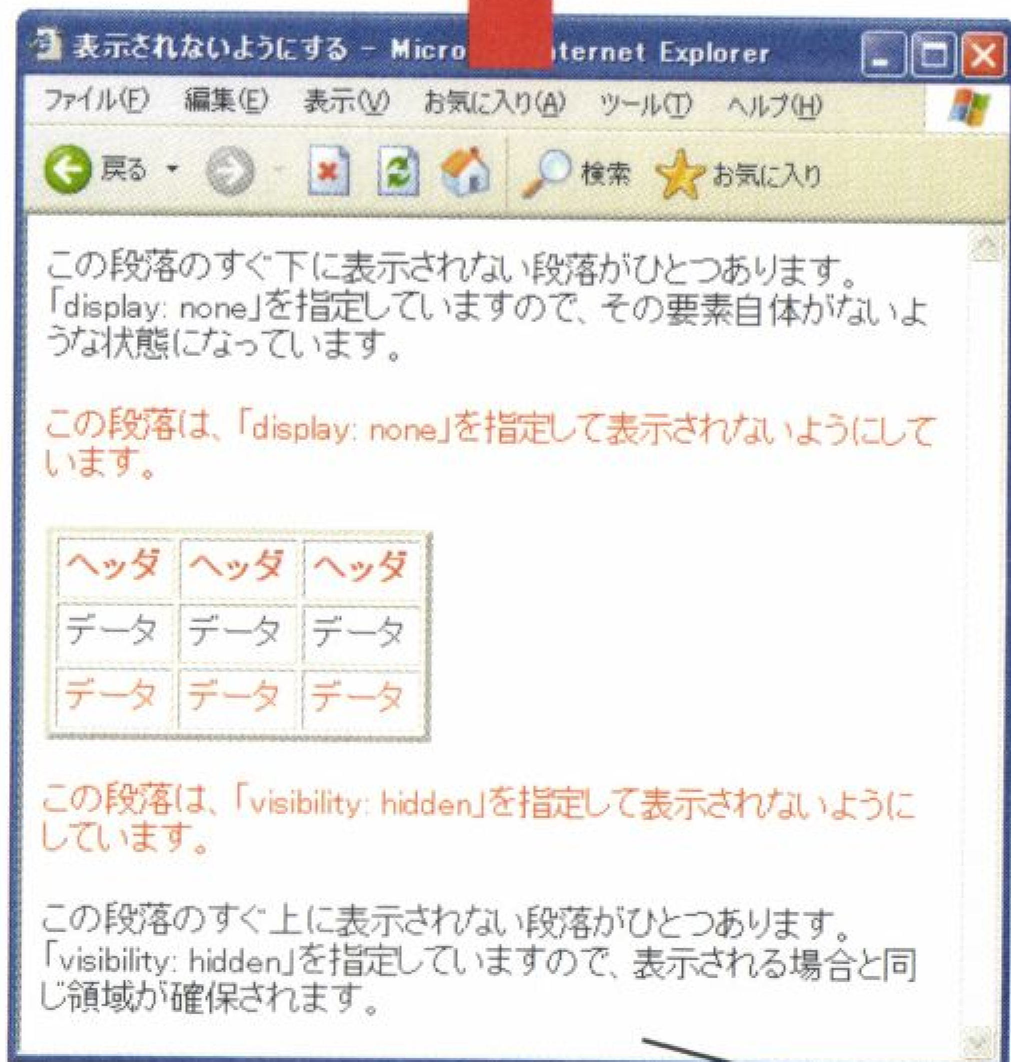
**display: none** ←その要素がない状態にする
**visibility: hidden** ←領域は確保するが見えない状態にする

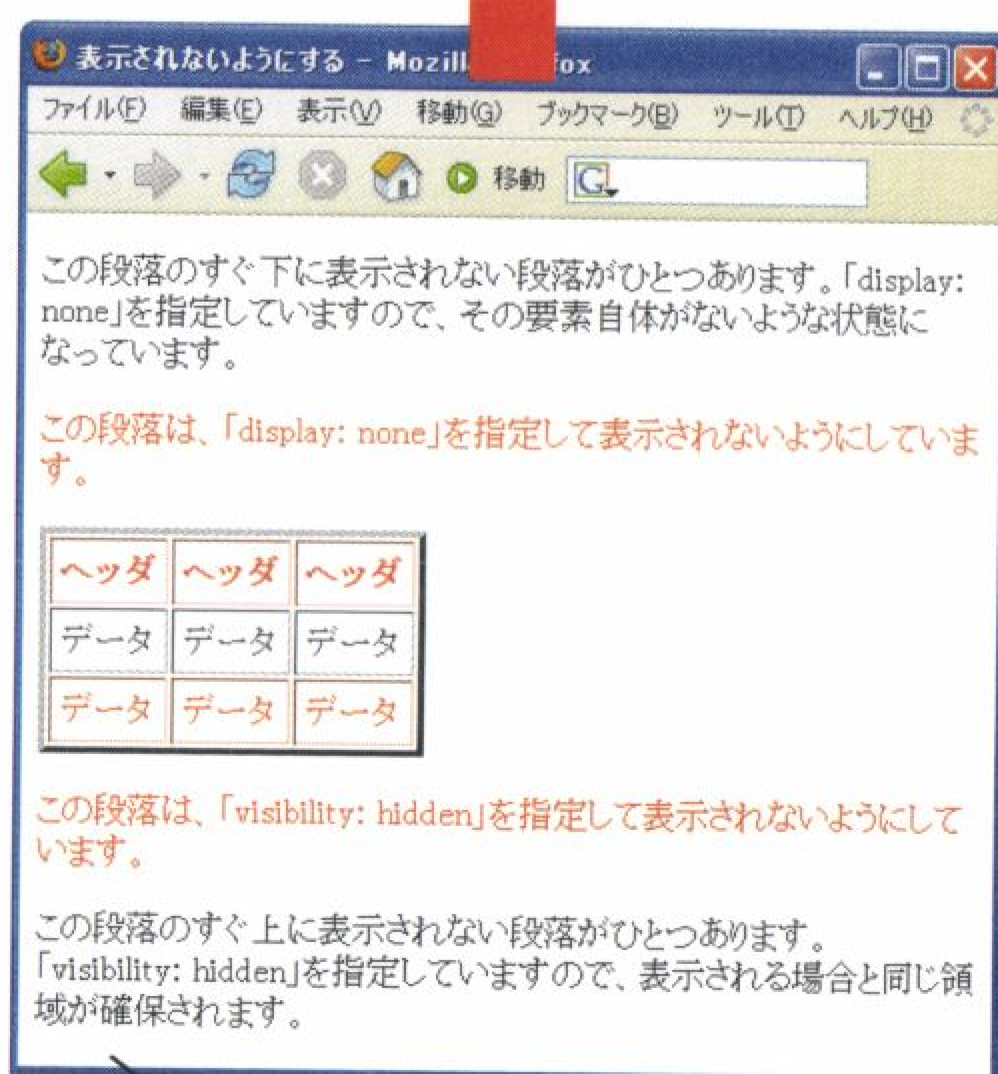
Internet Explorer 6.0




Firefox 1.0




すべてを表示させた状態

これらの用法は、指定された要素が表示されないようにします。
「display: none」を指定するとボックスそのものが生成されなくなり、あたかもその要素がないような状態になります。「visibility: hidden」を指定すると、その要素の表示領域は確保されますが、見えない状態(つまり、その要素が透明になったような状態)になります。

## Sample

### 【CSS】

```
.none { display: none }
.hidden { visibility: hidden }
```

### 【HTML】

```
<p>
この段落のすぐ下に表示されない段落がひとつあります。「display: none」を指定していますので、その要素自体がないような状態になっています。
</p>

<p class="none">
この段落は、「display: none」を指定して表示されないようにしています。
</p>

<table border="3" cellpadding="4">
<tr class="hidden">
<th>ヘッダ</th><th>ヘッダ</th><th>ヘッダ</th>
</tr>
<tr>
<td>データ</td><td>データ</td><td>データ</td>
</tr>
<tr class="hidden">
<td>データ</td><td>データ</td><td>データ</td>
</tr>
</table>

<p class="hidden">
この段落は、「visibility: hidden」を指定して表示されないようにしています。
</p>

<p>
この段落のすぐ上に表示されない段落がひとつあります。「visibility: hidden」を指定していますので、表示される場合と同じ領域が確保されます。
</p>
```

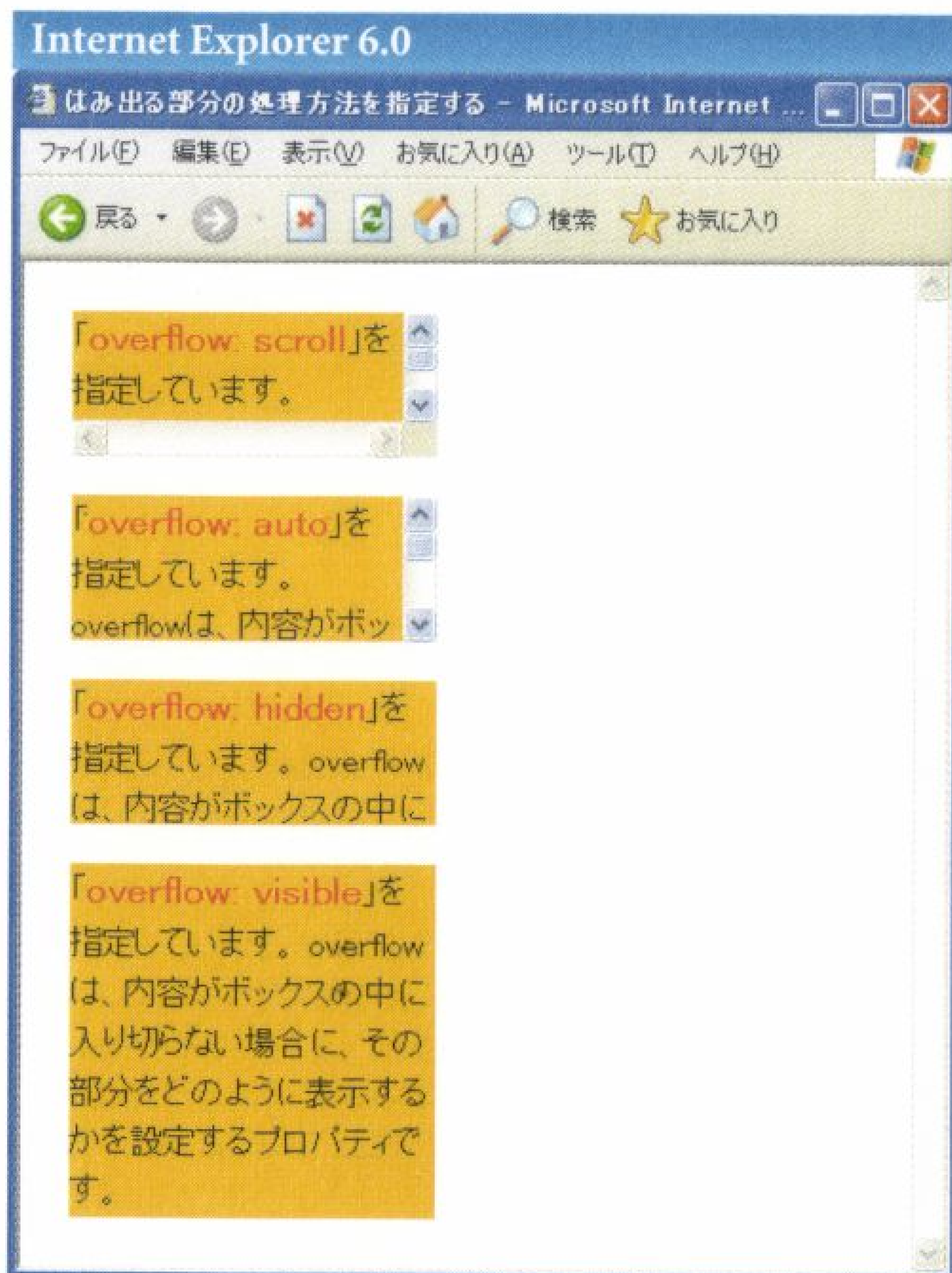
CSS

# はみ出る部分の処理方法を指定する

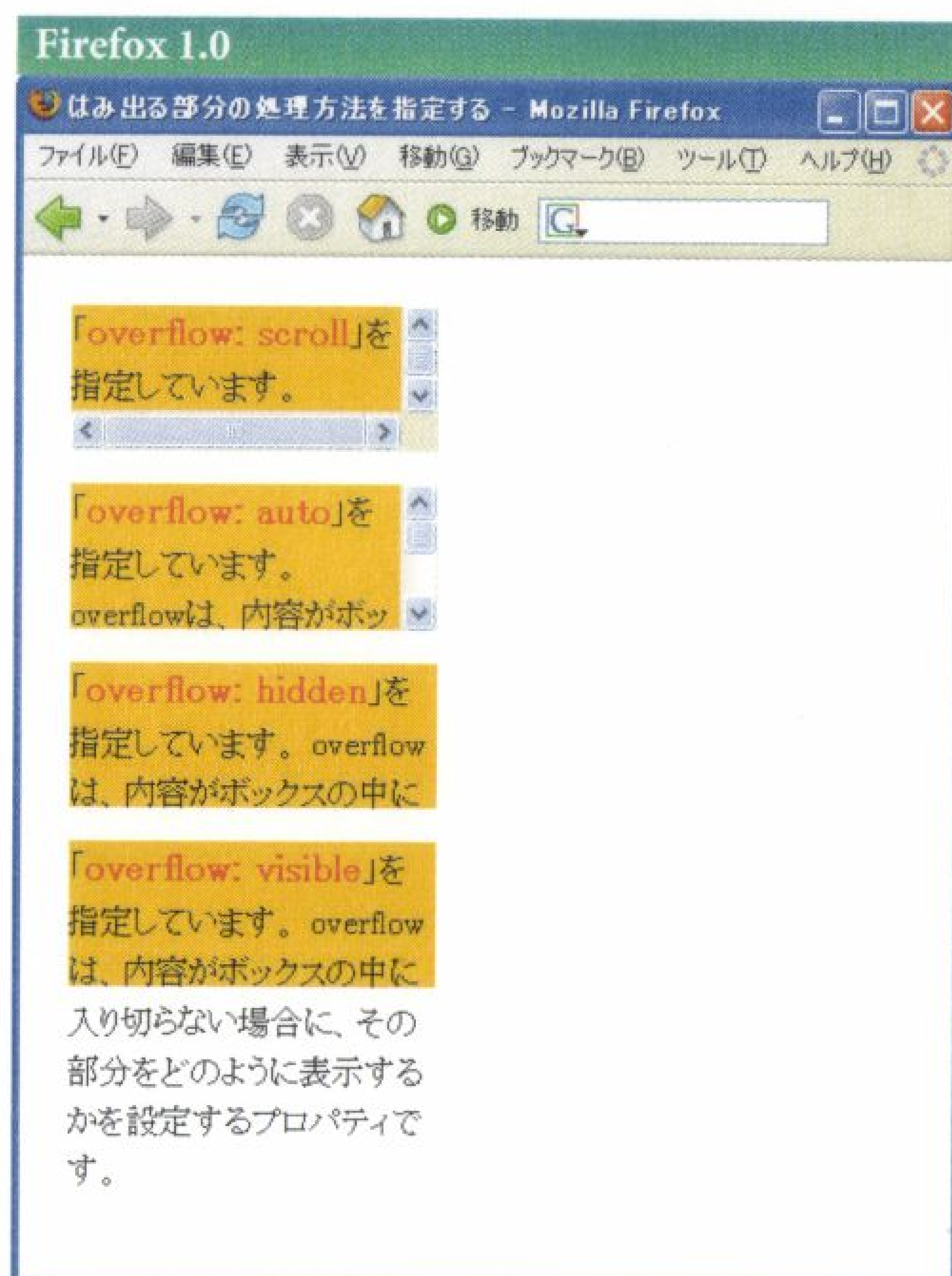
## overflow: 表示形式

| 【表示形式】 | |
|---|---|
| visible | ボックスからはみ出して表示 |
| hidden | はみ出した部分を表示しない |
| scroll | スクロールして見られるようにする |
| auto | 必要に応じてスクロールできるようにする |

Internet Explorer 6.0




Firefox 1.0




overflowプロパティは、内容がボックスに入り切らない場合に、その部分をどのように処理するかを指定します。
このプロパティは、ブロックレベル要素と置換要素(imgやinput、textare、selectなど)に対して指定することができます。

## Sample

### 【CSS】

```
body { margin: 1.5em }
#sample1 { overflow: scroll }
#sample2 { overflow: auto }
#sample3 { overflow: hidden }
#sample4 { overflow: visible }
p {
  width: 180px;
  height: 70px;
```

```
  line-height: 1.5;
  color: #000000;
  background: #ffcc00
}
em {
  font-style: normal;
  font-weight: bold;
  font-size: large;
  color: #ff0000;
  background: transparent
}
```

**【HTML】**

```
<p id="sample1">
「<em>overflow: scroll</em>」を指定しています。
overflowは、内容がボックスの中に入り切らない場合に、その部分をどのように表示するかを設定するプロパティです。
</p>

<p id="sample2">
「<em>overflow: auto</em>」を指定しています。
overflowは、内容がボックスの中に入り切らない場合に、その部分をどのように表示するかを設定するプロパティです。
</p>

<p id="sample3">
「<em>overflow: hidden</em>」を指定しています。
overflowは、内容がボックスの中に入り切らない場合に、その部分をどのように表示するかを設定するプロパティです。
</p>

<p id="sample4">
「<em>overflow: visible</em>」を指定しています。
overflowは、内容がボックスの中に入り切らない場合に、その部分をどのように表示するかを設定するプロパティです。
</p>
```


コラム「表のヘッダとフッタを固定して内容をスクロールさせる」(P.91)

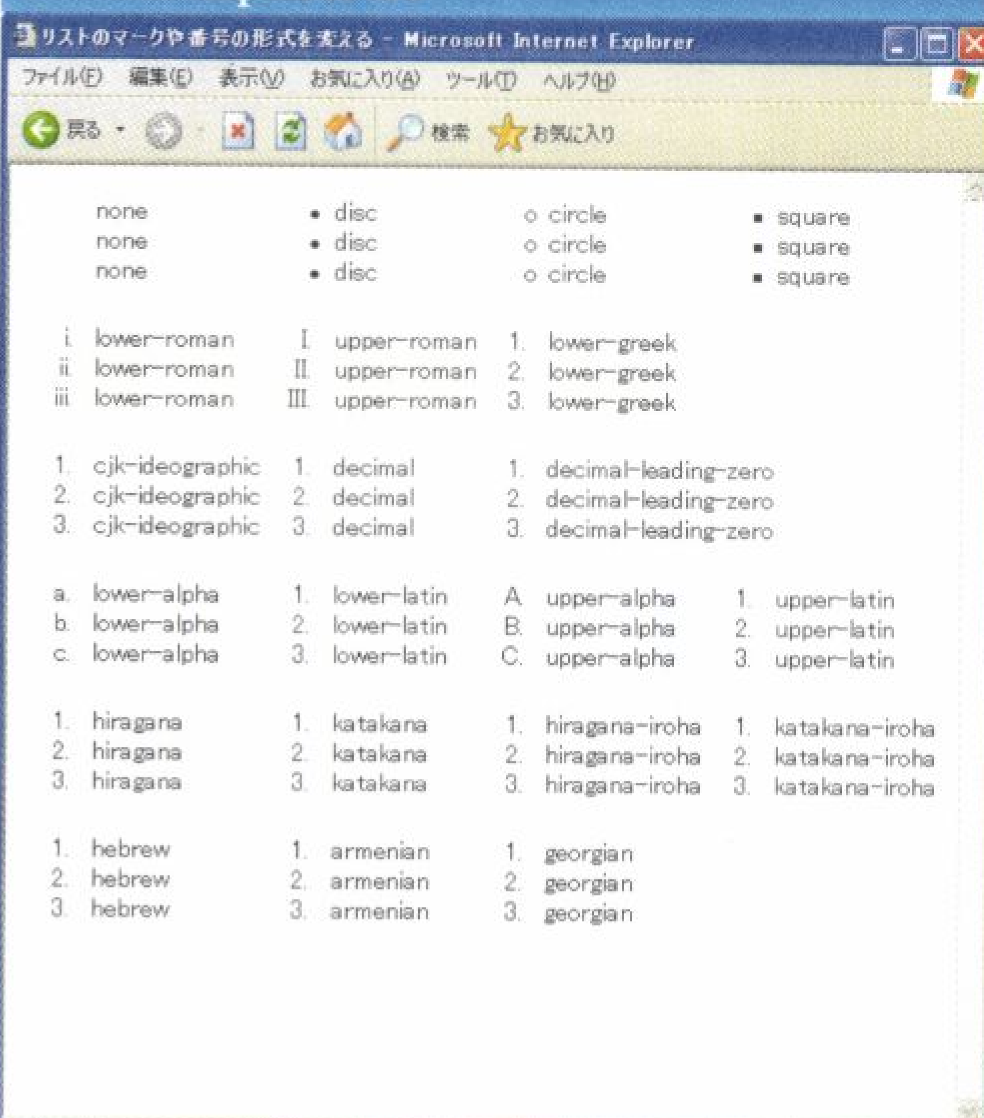
CSS

# リストのマークや番号の形式を変える

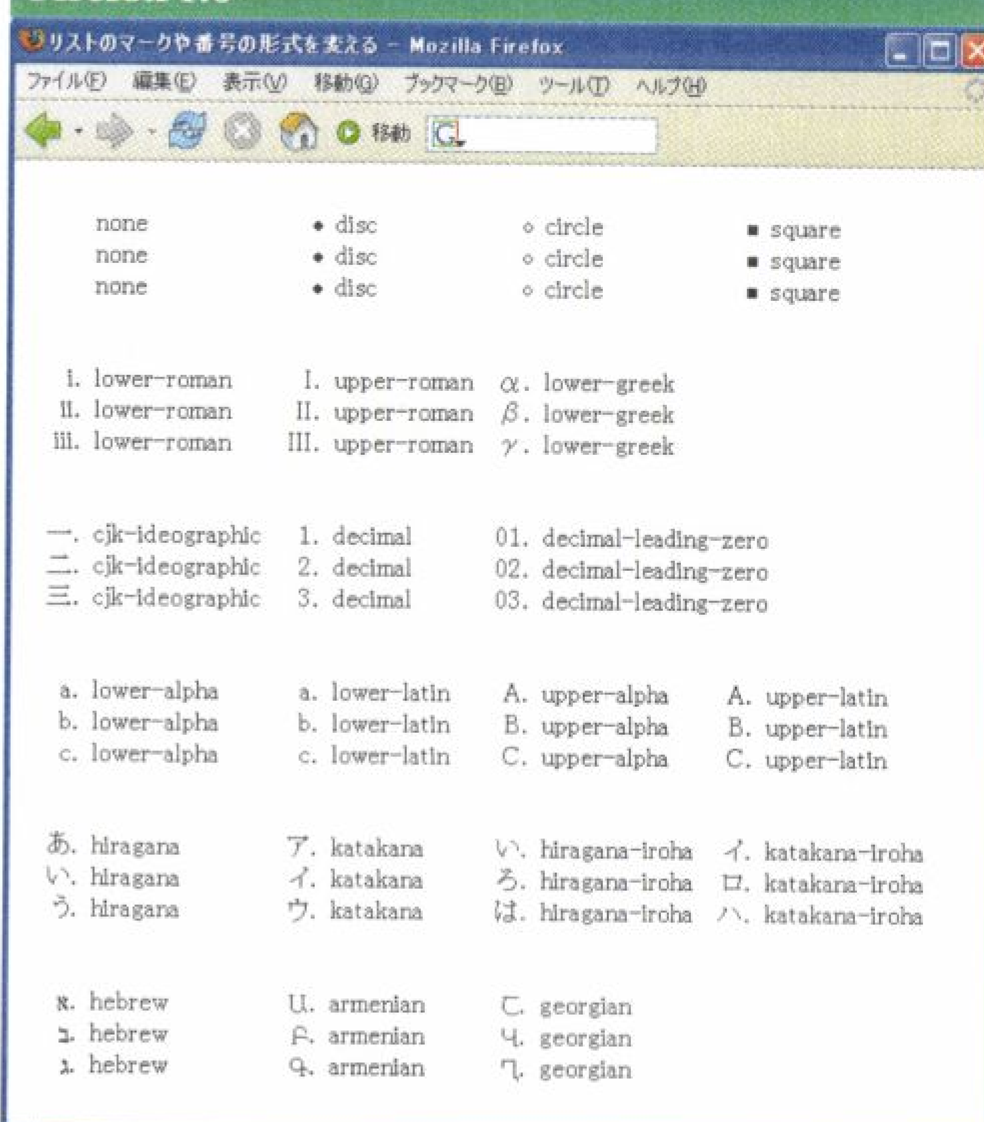
## list-style-type: *種類*

| 【種類】 | |
|---|---|
| none | 表示しない |
| disc | 塗りつぶされた丸（初期値） |
| circle | 線で描かれた丸 |
| square | 線で描かれた四角 |
| lower-roman | ローマ数字の小文字 |
| upper-roman | ローマ数字の大文字 |
| lower-greek | ギリシャ文字の小文字 |
| decimal | 算用数字 |
| decimal-leading-zero | 頭に0を付けた算用数字 |
| lower-latin | アルファベットの小文字 |
| lower-alpha | アルファベットの小文字 |
| upper-latin | アルファベットの大文字 |
| upper-alpha | アルファベットの大文字 |
| cjk-ideographic | 漢数字 |
| hiragana | ひらがなで「あいうえお」順 |
| katakana | カタカナで「アイウエオ」順 |
| hiragana-iroha | ひらがなで「いろは」順 |
| katakana-iroha | カタカナで「イロハ」順 |
| hebrew | ヘブライ数字 |
| armenian | アルメニア数字 |
| georgian | グルジア数字 |

Internet Explorer 6.0




Firefox 1.0



list-style-typeは、リストのマークや番号の形式を設定します。
list-style-imageプロパティで画像が指定されている場合には、その画像が優先して表示されます。

**Sample**

```
ul { list-style-type: disc }
```

**コラム**

### リストのアクセシビリティ

音声ブラウザなどでリストが表現される場合、リストに番号が付いていると、各項目の区切りが明確になってわかりやすくなります。しかし、その場合でもリストが入れ子になっている（リストの中にさらにリストを入れる）と、逆に混乱する可能性もあります。リストは一般的にその階層をインデントで表現しますが、それが見えない場合には階層が変化していることを認識できないためです。

CSS2の仕様では、「1.1」「1.2」「1.3」のように番号で階層を示す方法も提供されているのですが、ほとんどのブラウザはまだ対応していません。現状では、音声でもリストの階層がわかるようにするためには、各項目の内容自体を工夫するか、音声用に各階層の始めと終わりがわかるような目印を入れる（通常は「display: none」で表示させないようにする）などの対処が必要になります。現実的に考えると、リストはできるだけ単純にして、入れ子にすることは避けたほうがよいでしょう。

CSS

# リストのマークを画像にする

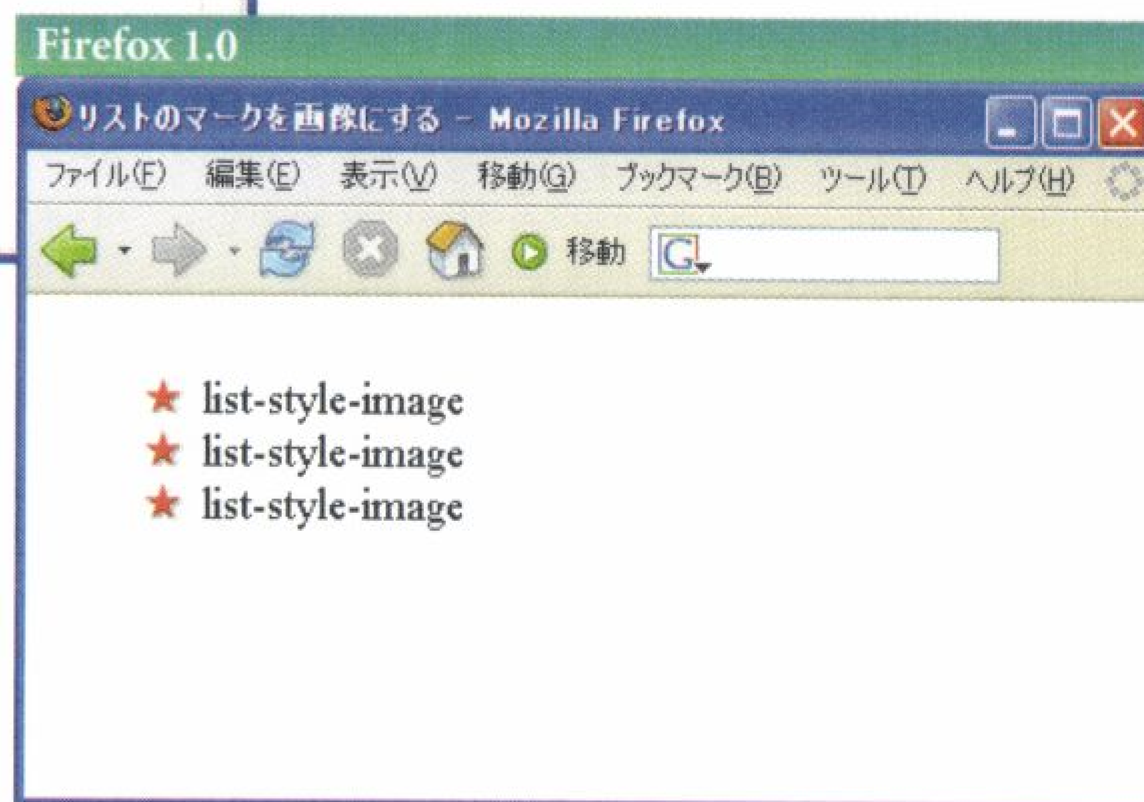
## list-style-image: url(*URL*)

| | |
|---|---|
| URL | 画像のURL |

Internet Explorer 6.0




Firefox 1.0




list-style-imageプロパティは、リストのマークとして表示させたい画像を指定します。list-style-typeプロパティが同時に設定されている場合でも、画像が優先して表示されます。

### Sample

**【CSS】**

```
ul {
  list-style-image: url(star.gif);
  font: large "Times New Roman", Times, serif
}
```

**【HTML】**

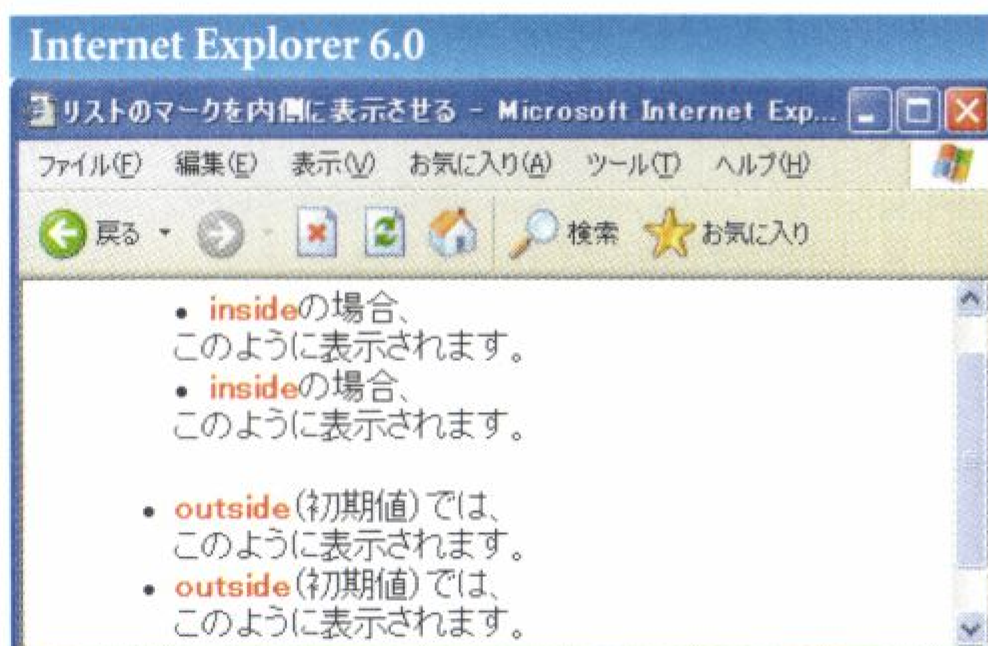
```
<ul>
<li>list-style-image</li>
<li>list-style-image</li>
<li>list-style-image</li>
</ul>
```

# リストのマークを内側に表示させる

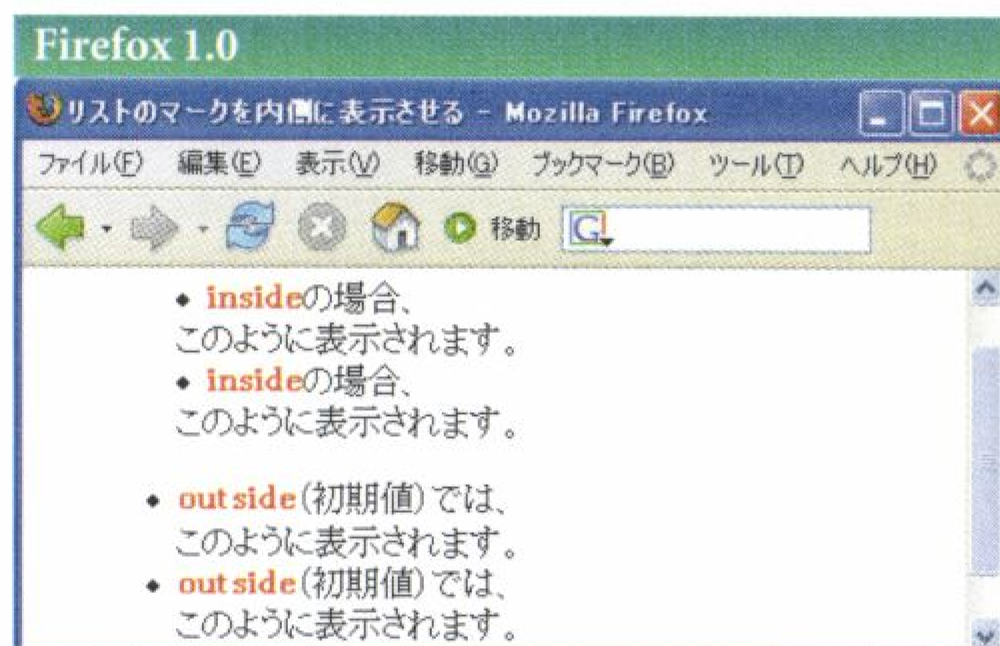
## list-style-position: 表示位置

| 【表示位置】 | |
|---|---|
| outside | マークを外側に表示（初期値） |
| inside | マークを内側に表示 |

Internet Explorer 6.0




Firefox 1.0




list-style-position プロパティは、リストのマークをリスト項目の表示領域の外側に表示するか、内側に表示するかを指定します。

### Sample

**【CSS】**

```
.in { list-style-position: inside }
em {
  color: #ff3300;
  background-color: #ffffff;
  font-style: normal;
  font-weight: bold
}
```

**【HTML】**

```
<ul class="in">
<li><em>inside</em>の場合、<br>このように表示されます。
</li>
<li><em>inside</em>の場合、<br>このように表示されます。
</li>
</ul>

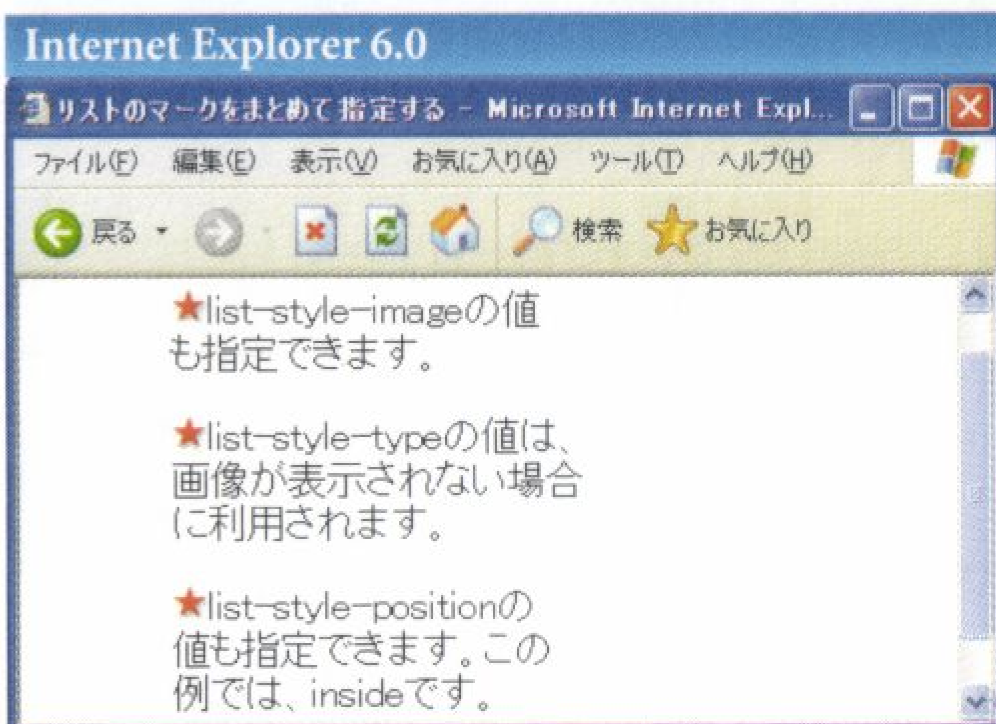
<ul>
<li><em>outside</em>（初期値）では、<br>このように表示されます。
</li>
<li><em>outside</em>（初期値）では、<br>このように表示されます。
</li>
</ul>
```

# リストのマークをまとめて指定する

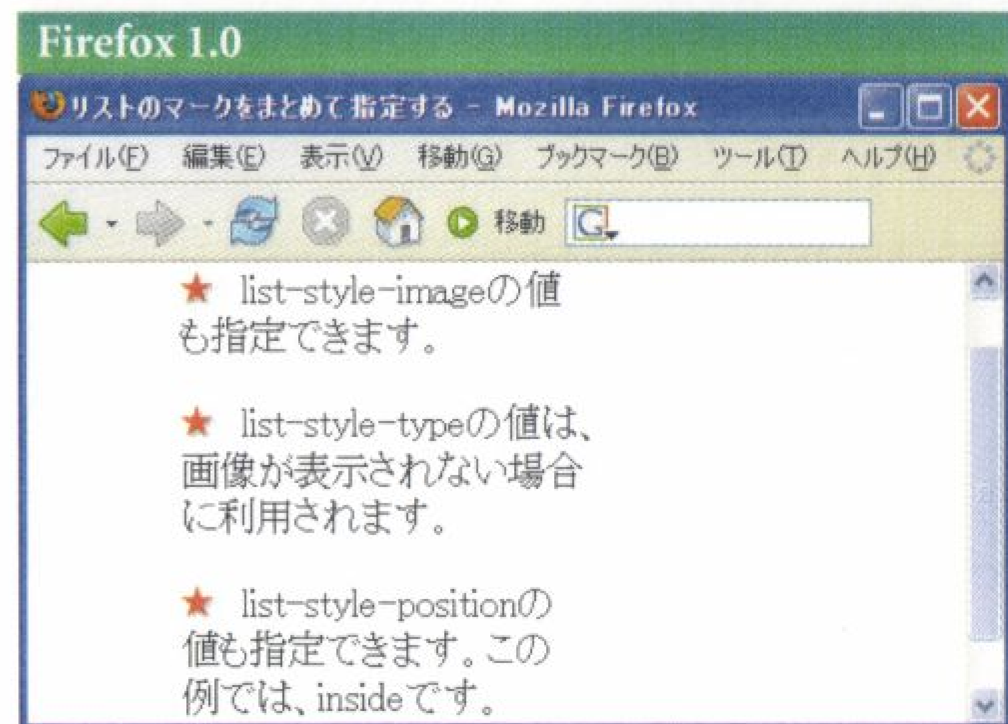
## list-style: リスト関連のプロパティの値

| 【リスト関連のプロパティの値】 |
| --- |
| list-style-type (P.270) で指定できる値 |
| list-style-image (P.272) で指定できる値 |
| list-style-position (P.273) で指定できる値 |

Internet Explorer 6.0




Firefox 1.0




これらのプロパティは、リストのマークに関連するプロパティの値をまとめて指定します。

必要な値を任意の順序で半角スペースで区切って指定します。値として「none」を指定すると、マークが表示されなくなります。

### Sample

**【CSS】**

```
ul {
  font-size: large;
  list-style: url(star.gif) disc inside
}
li { margin-bottom: 1em }
```

**【HTML】**

```
<ul>
<li>
list-style-imageの値<br>も指定できます。
</li>
<li>
list-style-typeの値は、<br>画像が表示されない場合<br>に利用されます。
</li>
<li>
list-style-positionの<br>値も指定できます。この<br>例では、insideです。
</li>
</ul>
```

# 表の枠線を単一の線にする

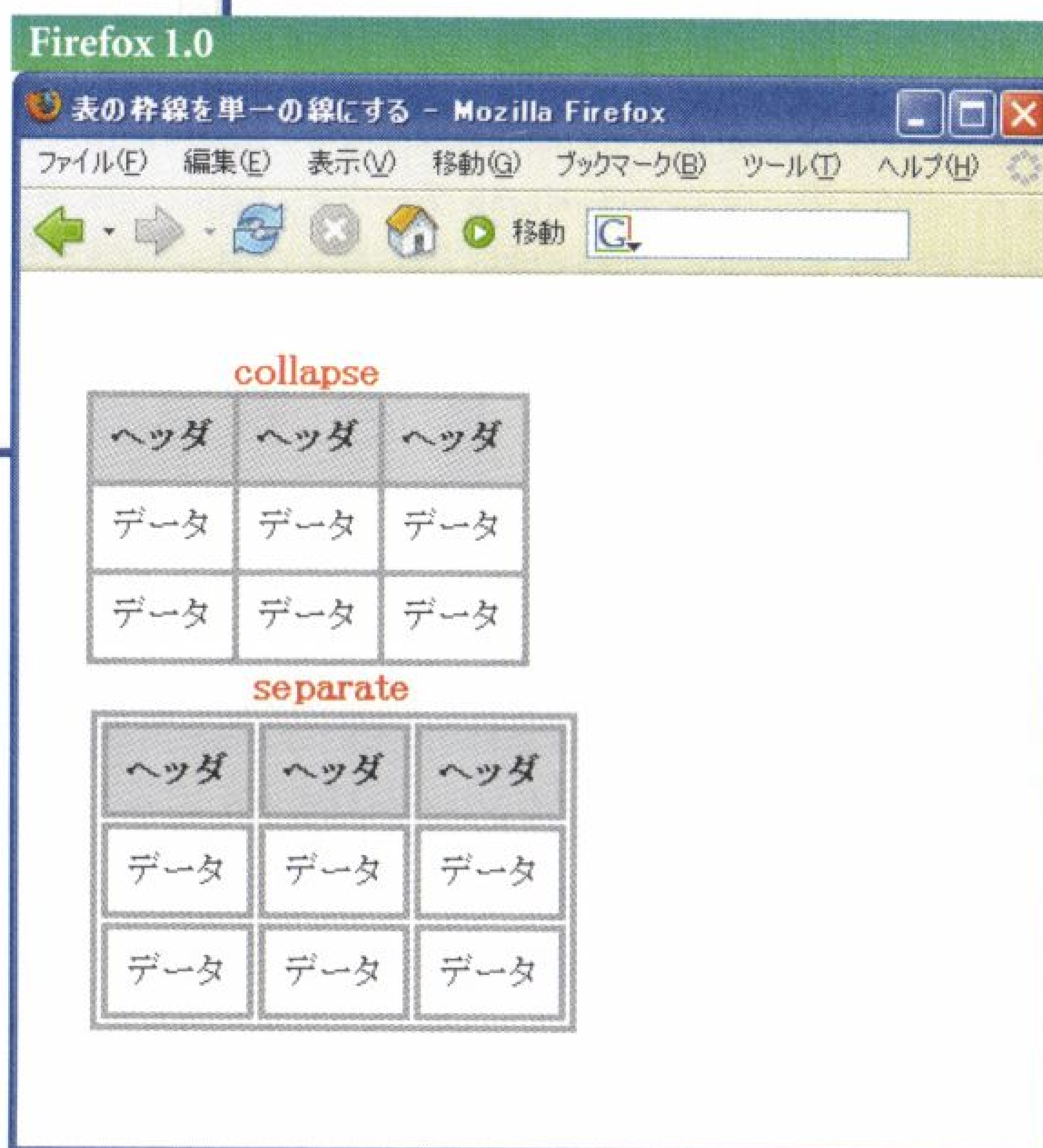
## border-collapse: 表の枠線の表示形式

| 【表の枠線の表示形式】 | |
|---|---|
| collapse | 表の外枠や各セルの枠を重ねて表示 |
| separate | 表の外枠や各セルの枠を別に表示 |

Internet Explorer 6.0




Firefox 1.0




border-collapse プロパティは、表の外枠や各セルの枠線を重ねて(単一の線として)表示するか、別々に表示するかを指定します。
このプロパティは、table 要素に対してのみ指定できます。

### Sample

【CSS】

```
table#sample1 { border-collapse: collapse }
table#sample2 { border-collapse: separate }
```

```
table, th, td { border: 3px solid #999999 }
th {
  color: #000000;
  background-color: #cccccc
}
caption{
  font-size: large;
  font-weight: bold;
  color: #ff3300;
  background: transparent
}
```

**【HTML】**

```
<table border="3" cellpadding="8" id="sample1">
<caption>collapse</caption>
<tr><th>ヘッダ</th><th>ヘッダ</th><th>ヘッダ</th></tr>
<tr><td>データ</td><td>データ</td><td>データ</td></tr>
<tr><td>データ</td><td>データ</td><td>データ</td></tr>
</table>

<table border="3" cellpadding="8" id="sample2">
<caption>separate</caption>
<tr><th>ヘッダ</th><th>ヘッダ</th><th>ヘッダ</th></tr>
<tr><td>データ</td><td>データ</td><td>データ</td></tr>
<tr><td>データ</td><td>データ</td><td>データ</td></tr>
</table>
```

**コラム**

### セル同士の異なる枠線が重なった場合の優先順位は？

border-collapseプロパティの値を「collapse」にすると、表の枠線が重なって単一の線として表示されますが、もし太さや形式などの異なる枠線が重なった場合にはどのように表示されるのでしょうか？　CSS2の仕様では、異なる種類の枠線が重なった場合の表示の優先順位を以下のように規定しています。

1. border-styleプロパティの値が「hidden」のものは最優先される
2. border-styleプロパティの値が「none」のものは優先度がもっとも低い
3. 「hidden」と「none」以外が指定されている場合、より太い枠線が優先される。太さが同じで形式が異なる場合の優先順位は、以下の通り（優先度が高い順）

   double←solid←dashed←dotted←ridge←outset←groove←inset

4. 枠線の太さも形式も同じ場合の優先順位は、以下の通り（優先度が高い順）

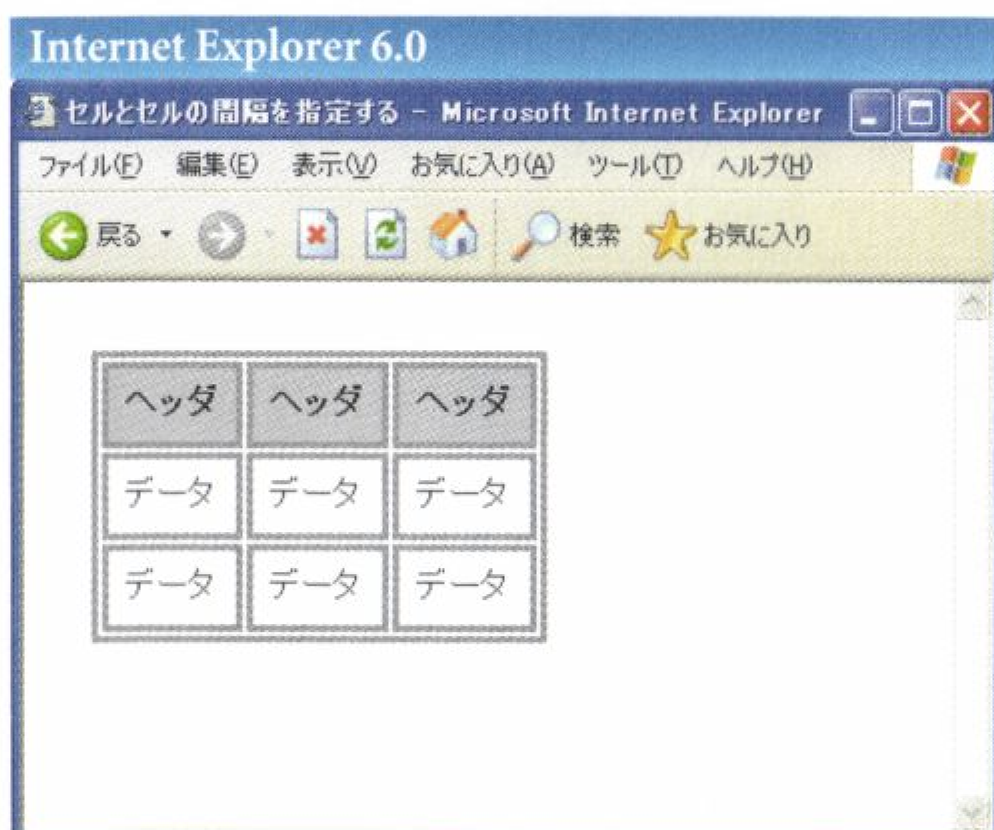
   th要素・td要素←tr要素←thead・tbody・tfoot要素←col要素←colgroup要素←table要素

# セルとセルの間隔を指定する

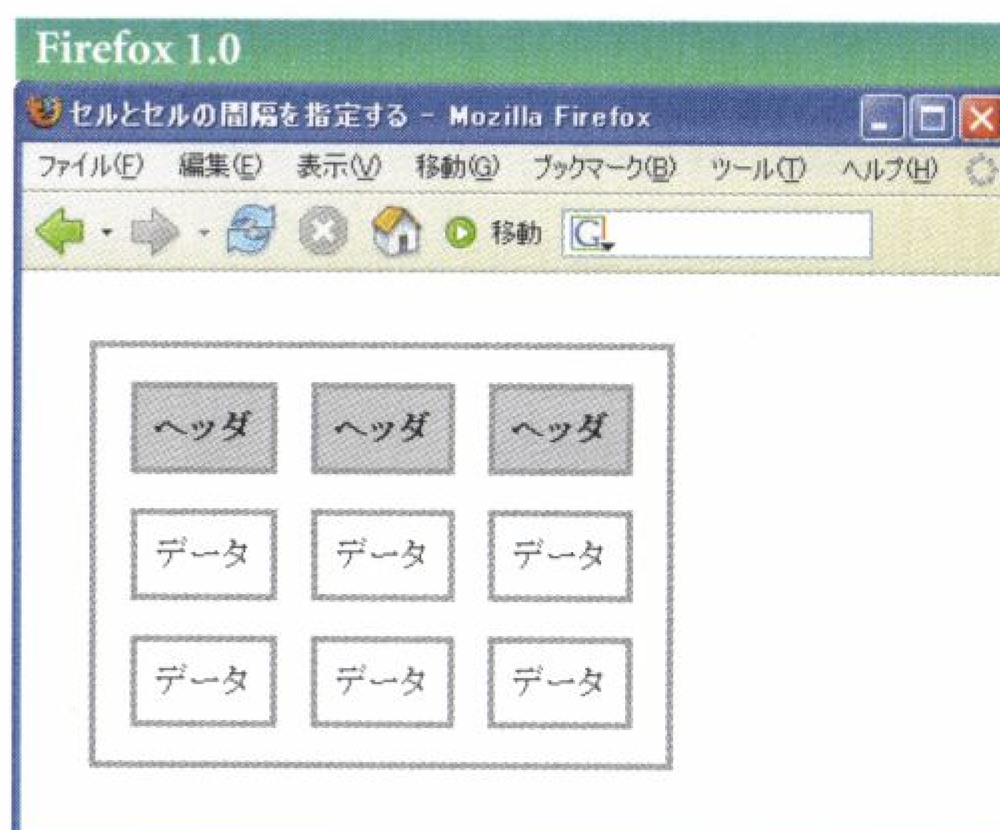
**border-spacing: *間隔***

| 間隔 | 単位付きの数値 |
|---|---|

Internet Explorer 6.0




Firefox 1.0




border-spacing プロパティは、隣接する各セルの枠線と枠線の間隔を指定します。値は半角スペースで区切ってふたつ指定することもできますが、与えられた値の個数によって次のように設定されます。

| | |
|---|---|
| ・値が１つの場合 | 値１→上下左右の間隔 |
| ・値が２つの場合 | 値１→左右の間隔　値２→上下の間隔 |

## Sample

**【CSS】**

```
table { border-spacing: 1em }
table, th, td { border: 3px solid #999999 }
th {
  color: #000000;
  background-color: #cccccc
}
```

**【HTML】**

```
<table border="3" cellpadding="8">
<tr><th>ヘッダ</th><th>ヘッダ</th><th>ヘッダ</th></tr>
<tr><td>データ</td><td>データ</td><td>データ</td></tr>
<tr><td>データ</td><td>データ</td><td>データ</td></tr>
</table>
```

# 表のタイトルを下に表示させる

## caption-side: 配置位置

| 配置位置 | top・bottom |
|---|---|



Internet Explorer 6.0




Firefox 1.0




caption-sideプロパティは、表のタイトル(caption要素)の表示位置を指定します。「top」を指定すると表の上に、「bottom」を指定すると表の下に表示します。仕様上は「left」と「right」も指定できますが、正しく対応しているブラウザはほとんどないようです。

## Sample

**【CSS】**

```
caption {
  caption-side: bottom;
  font-size: large;
  font-weight: bold;
  color: #ff3300;
  background: transparent
}
table, th, td { border: 3px solid #999999 }
th {
  color: #000000;
  background-color: #cccccc
}
```

**【HTML】**

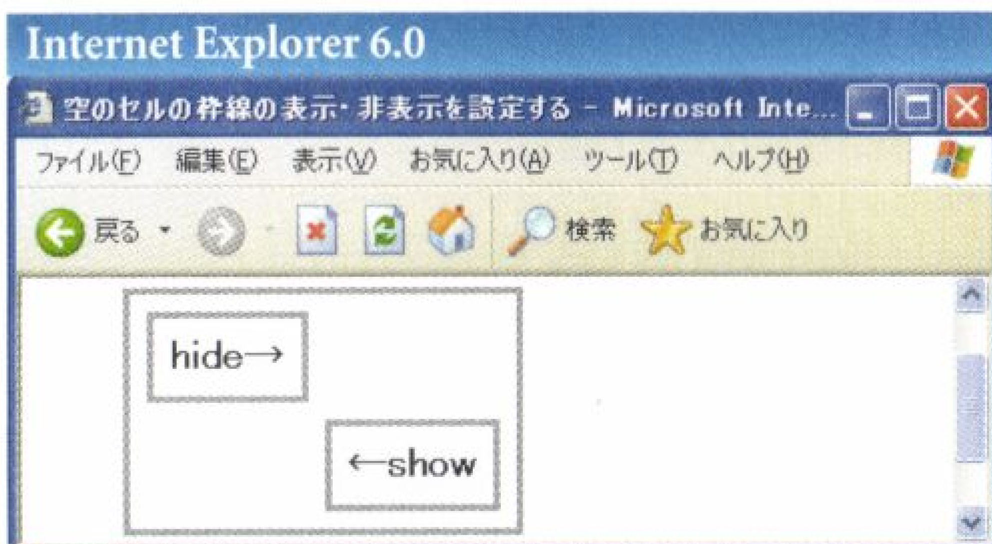
```
<table border="3" cellpadding="8">
<caption>caption-side: bottom</caption>
<tr><th>ヘッダ</th><th>ヘッダ</th><th>ヘッダ</th></tr>
<tr><td>データ</td><td>データ</td><td>データ</td></tr>
<tr><td>データ</td><td>データ</td><td>データ</td></tr>
</table>
```

# 空のセルの枠線の表示・非表示を設定する

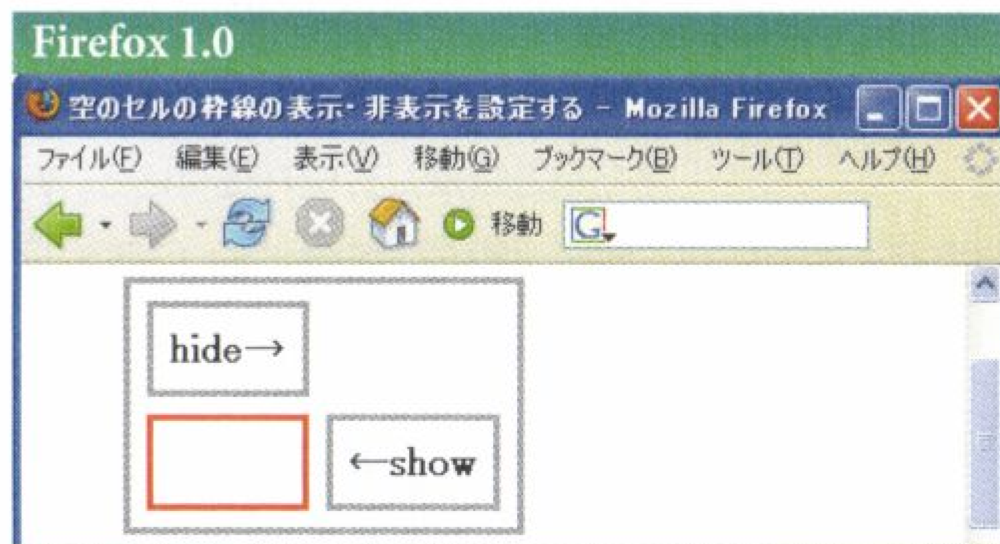
## empty-cells: 枠線を表示するかどうか

枠線を表示するかどうか　　show・hide

Internet Explorer 6.0




Firefox 1.0




empty-cellsプロパティは、空のセルの枠線を表示させるかどうかを指定します。この場合の「空のセル」とは、内容が空であるセルだけでなく、visibilityプロパティの値が「hidden」に設定されているセルも含みます。値として「show」を指定すると枠線を表示し、「hide」を指定すると枠線を表示しなくなります。

## Sample

【CSS】

```
table, td { border: 3px solid #999999 }
td {
  font-size: large;
  font-weight: bold
}
td.hide { empty-cells: hide }
td.show {
  empty-cells: show;
  border-color: #ff3300
}
```

【HTML】

```
<table border="3" cellpadding="8" cellspacing="8">
<tr>
  <td>hide→</td>
  <td class="hide"></td>
</tr>
<tr>
  <td class="show"></td>
  <td>←show</td>
</tr>
</table>
```


visibility: hidden：「表示と配置」の「表示されないようする」(P.266)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X Opera7 Safari IE5-mac IE4-mac

# カーソルの形を指定する

**cursor:** *形状*
**cursor: url(*URL*)**

| | |
|---|---|
| 形状 | auto・crosshair・default・pointer・move・text・wait・help・e-resize・ne-resize・nw-resize・n-resize・se-resize・sw-resize・s-resize・w-resize |
| URL | カーソルのURL |

### Windowsでの表示例

| | | | |
|---|---|---|---|
| | | ⇧ | n-resize |
| ＋ | crosshair | ⇩ | s-resize |
| | default | ⇦ | w-resize |
| | pointer | ⇨ | e-resize |
| | move | | ne-resize |
| I | text | | nw-resize |
| | wait | | se-resize |
| ? | help | | sw-resize |

### Macintoshでの表示例

| | | | |
|---|---|---|---|
| | | | n-resize |
| ＋ | crosshair | | s-resize |
| | default | | w-resize |
| | pointer | | e-resize |
| | move | | ne-resize |
| I | text | | nw-resize |
| | wait | | se-resize |
| ? | help | | sw-resize |

cursorプロパティは、マウスなどのポインティングデバイスのカーソルが、その要素の上にある時のカーソルの形状を設定します。
Internet Explorer 6.0以降では、自作のカーソル(「.cur」または「.ani」形式)を指定することもできます。

## Sample

```
code { cursor: text }
```

その他の設定をする

# 印刷時の改ページを指定する

**page-break-before: always**
**page-break-after: always**



page-break-before プロパティと page-break-after プロパティは、印刷時に指定した部分の前または後で改ページをするように設定します。

「page-break-before: always」は指定した要素の前で、「page-break-after: always」は指定した要素の後で改ページします。指定できる値としては、仕様上は他にも用意されているのですが、現在利用できるのはこの値だけのようです。このプロパティは印刷時にのみ有効で、画面表示には影響を与えません。

## Sample

```
h1, table { page-break-before: always }
```

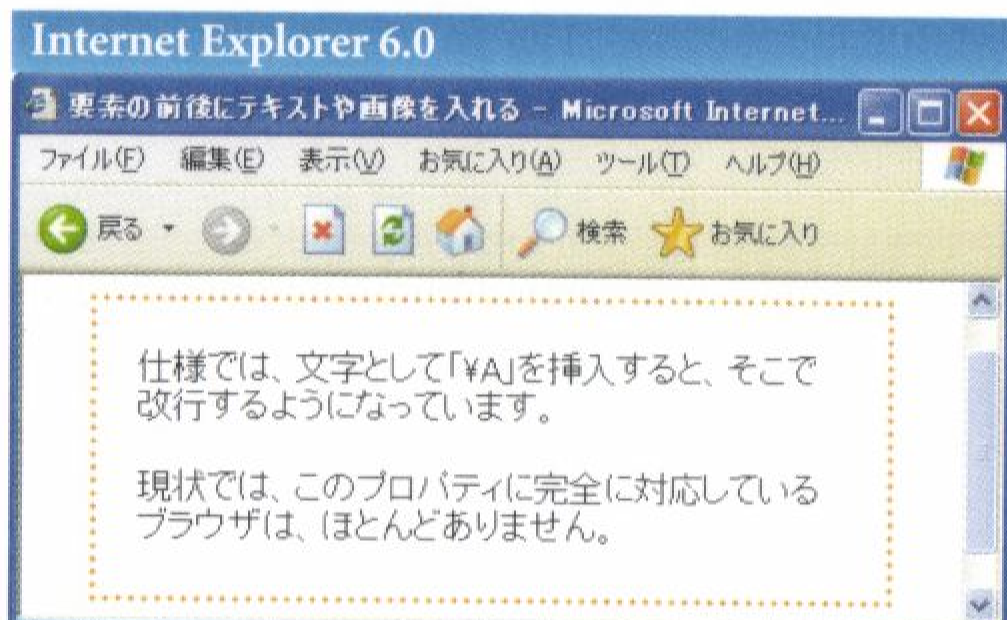
# 要素の前後にテキストや画像を入れる

要素名:before { content: "テキスト" } ←直前にテキストを入れる
要素名:before { content: url(URL) } ←直前に画像を入れる
要素名:after { content: "テキスト" } ←直後にテキストを入れる
要素名:after { content: url(URL) } ←直後に画像を入れる

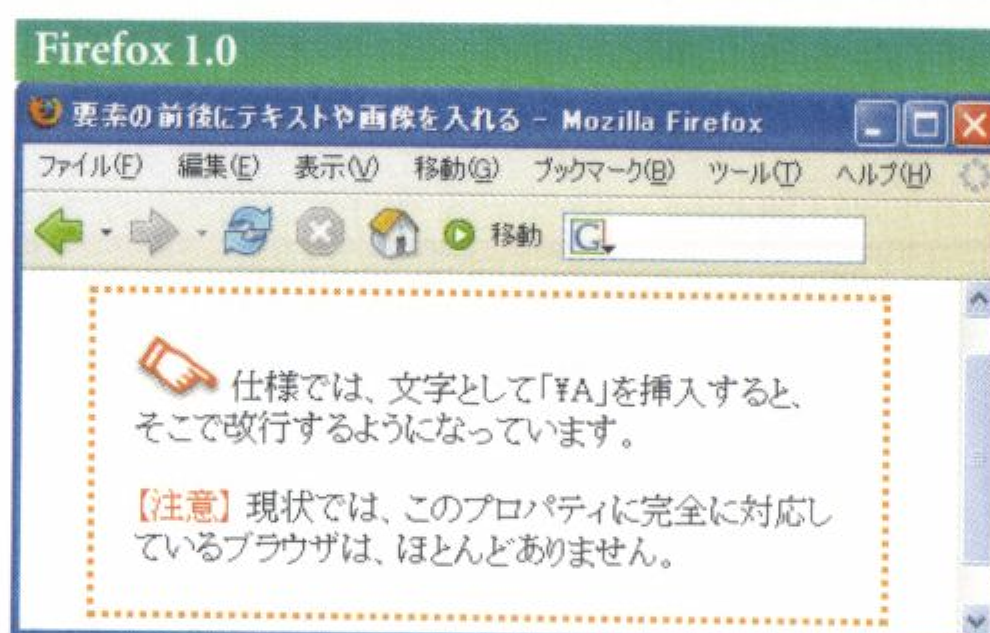
※「要素名」の部分には、「#ID名」や「.クラス名」も指定できます。

URL　　画像のURL

Internet Explorer 6.0




Firefox 1.0




「:before」や「:after」は、「要素名」で指定した部分の直前または直後に、テキストや画像を挿入します。

## Sample

**【CSS】**

```
.note:before { content: url(hand.gif) }
.warning:before {
  content: "【注意】";
  color: #ff0000;
  background-color: #ffffff
}
div.info {
  border: dotted 3px #ff9900;
  padding: 0.2em 1.2em
}
```

**【HTML】**

```
<div class="info">
<p class="note">
仕様では、文字として「¥A」を挿入すると、そこで改行するようになっています。
</p>
<p class="warning">
現状では、このプロパティに完全に対応しているブラウザは、ほとんどありません。
</p>
</div>
```

# 引用符として使用する記号を設定する

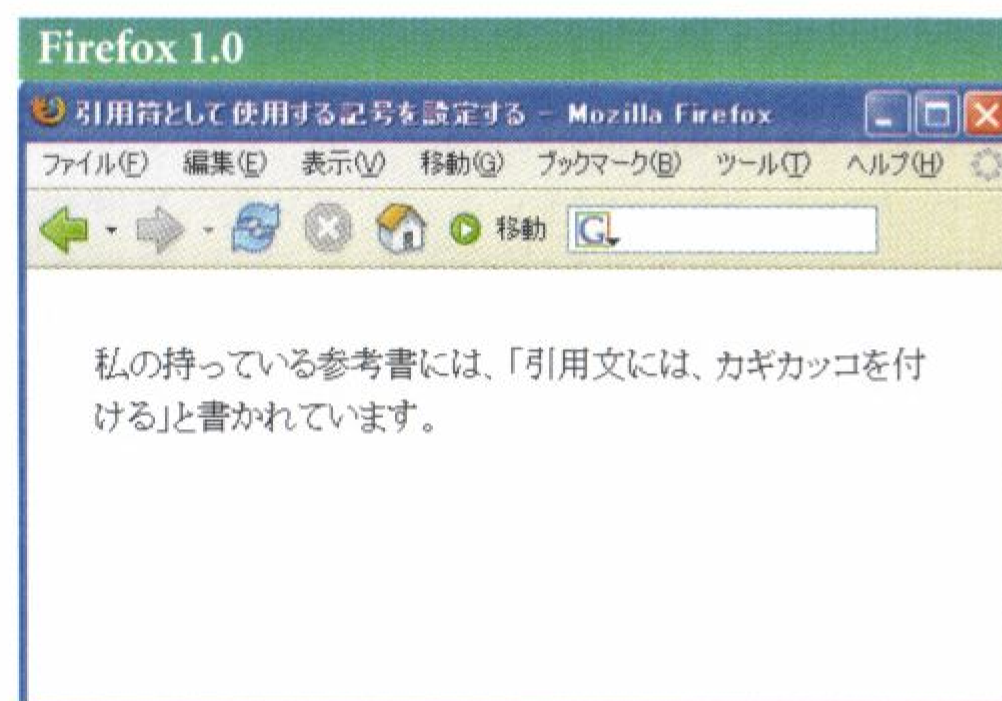
**q { quotes: "*記号1*" "*記号2*" }** ←引用符の設定
**q:before { content: open-quote }** ←引用符の追加（前）
**q:after { content: close-quote }** ←引用符の追加（後）

| | |
|---|---|
| 記号1 | 引用部分の前に付ける記号 |
| 記号2 | 引用部分の後に付ける記号 |

Internet Explorer 6.0




Firefox 1.0




これらの用法は、短い引用文を示すq要素の前後に付ける引用符を設定します。quotesプロパティで引用符として使用する記号を設定し、「q:before ～」と「q:after ～」の書式でその引用符をq要素の前後に追加します。この時、デフォルトの「"」などの引用符は、指定した記号に置き換えられます。

## Sample

**【CSS】**

```
body {
  margin: 2em;
  line-height: 1.5
}
q { quotes: "「" "」" }
q:before { content: open-quote }
q:after  { content: close-quote }
```

**【HTML】**

```
<p>
私の持っている参考書には、<q>引用文には、カギカッコを付ける</q>と書かれています。
</p>
```

HTML：「テキスト」の「短い引用文を表す」(P.32)

CSS

# コメントを入れる

## /* コメント文 */

「/*～*/」は、CSSのソースの中にコメントを入れておく場合に使用します。コメントを入れ子にすることはできません。

### Sample

```
body { margin: 3em }        /* 上下左右のマージンを設定 */
h1 {
  font-size: medium;        /* フォントサイズを標準に */
  color: #ffffff;           /* 文字色を白に */
  background: #ff6600       /* 背景色をオレンジに */
}
p { line-height: 1.5 }      /* 段落の行間を通常の1.5倍に */
```

# CSS2全プロパティ一覧

| プロパティ名 | 説明 | 適用対象 | 初期値 | 継承 | 参照 |
|---|---|---|---|---|---|
| azimuth | 音の聞こえる横方向の位置 | すべての要素 | center | ○ | |
| background | 背景の一括指定 | すべての要素 | 個別に参照 | × | 236 |
| background-attachment | 背景画像の固定配置 | すべての要素 | scroll | × | 234 |
| background-color | 背景色 | すべての要素 | transparent | × | 224 |
| background-image | 背景画像 | すべての要素 | none | × | 227 |
| background-position | 背景画像の表示位置 | ブロックレベル・置換要素 | 0% 0% | × | 232 |
| background-repeat | 背景画像の並び方 | すべての要素 | repeat | × | 230 |
| border | 枠線の一括指定 | すべての要素 | 個別に参照 | × | 247 |
| border-collapse | セルの枠線の表示形式 | table要素 | collapse | ○ | 275 |
| border-color | 枠線の色の一括指定 | すべての要素 | 個別に参照 | × | 243 |
| border-spacing | セルとセルの間隔 | table要素 | 0 | ○ | 277 |
| border-style | 枠線の形式の一括指定 | すべての要素 | 個別に参照 | × | 245 |
| border-top | 上の枠線の色・形式・太さ | すべての要素 | 個別に参照 | × | 247 |
| border-bottom | 下の枠線の色・形式・太さ | すべての要素 | 個別に参照 | × | 247 |
| border-left | 左の枠線の色・形式・太さ | すべての要素 | 個別に参照 | × | 247 |
| border-right | 右の枠線の色・形式・太さ | すべての要素 | 個別に参照 | × | 247 |
| border-top-color | 上の枠線の色 | すべての要素 | colorの値 | × | 243 |
| border-bottom-color | 下の枠線の色 | すべての要素 | colorの値 | × | 243 |
| border-left-color | 左の枠線の色 | すべての要素 | colorの値 | × | 243 |
| border-right-color | 右の枠線の色 | すべての要素 | colorの値 | × | 243 |
| border-top-style | 上の枠線の形式 | すべての要素 | none | × | 245 |
| border-bottom-style | 下の枠線の形式 | すべての要素 | none | × | 245 |
| border-left-style | 左の枠線の形式 | すべての要素 | none | × | 245 |
| border-right-style | 右の枠線の形式 | すべての要素 | none | × | 245 |
| border-top-width | 上の枠線の太さ | すべての要素 | medium | × | 241 |
| border-bottom-width | 下の枠線の太さ | すべての要素 | medium | × | 241 |
| border-left-width | 左の枠線の太さ | すべての要素 | medium | × | 241 |
| border-right-width | 右の枠線の太さ | すべての要素 | medium | × | 241 |
| border-width | 枠線の太さの一括指定 | すべての要素 | 個別に参照 | × | 241 |
| bottom | 下からの位置 | 「position: static」以外の要素 | auto | × | 258, 260, 262 |
| caption-side | 表タイトルの表示位置 | caption要素 | top | ○ | 278 |
| clear | 回り込みの解除 | ブロックレベル要素 | none | × | 254 |
| clip | 見える範囲 | ブロックレベル・置換要素 | auto | × | |
| color | 文字色 | すべての要素 | ブラウザに依存 | ○ | 201 |
| content | 内容の追加 | :before・:after・「display: marker」の要素 | 空文字 | × | 282, 283 |

リファレンス

| プロパティ名 | 説明 | 適用対象 | 初期値 | 継承 | 参照 |
|---|---|---|---|---|---|
| counter-increment | 連番の値を進める | すべての要素 | none | × | |
| counter-reset | 連番のリセット | すべての要素 | none | × | |
| cue | 要素識別音の一括指定 | すべての要素 | 未定義 | × | |
| cue-after | 要素の直後に識別音を鳴らす | すべての要素 | none | × | |
| cue-before | 要素の直前に識別音を鳴らす | すべての要素 | none | × | |
| cursor | カーソルの形状 | すべての要素 | auto | ○ | 280 |
| direction | 文字表記の方向 | すべての要素 | ltr | ○ | |
| display | 表示形式 | すべての要素 | inline | × | 266 |
| elevation | 音の聞こえる縦方向の位置 | すべての要素 | level | ○ | |
| empty-cells | 空セルの枠の表示・非表示 | th・td要素 | show | ○ | 279 |
| float | 左右への配置と回り込み | すべての要素（例外あり） | none | × | 252 |
| font | フォントの一括指定 | すべての要素 | 個別に参照 | ○ | 210 |
| font-family | フォントの種類 | すべての要素 | ブラウザに依存 | ○ | 202 |
| font-size | フォントサイズ | すべての要素 | medium | ○ | 204 |
| font-size-adjust | フォントサイズの調整 | すべての要素 | none | ○ | |
| font-stretch | 長体・平体 | すべての要素 | normal | ○ | |
| font-style | イタリック | すべての要素 | normal | ○ | 208 |
| font-variant | スモールキャップ | すべての要素 | normal | ○ | |
| font-weight | フォントの太さ | すべての要素 | normal | ○ | 206 |
| height | ボックスの内容領域の高さ | テキストのインライン要素とcol・colgroup要素以外 | auto | × | 249 |
| left | 左からの位置 | 「position: static」以外の要素 | auto | × | 258, 260, 262 |
| letter-spacing | 文字間隔 | すべての要素 | normal | ○ | 217 |
| line-height | 行間 | すべての要素 | normal | ○ | 212 |
| list-style | リストのマークの一括指定 | li要素 | 未定義 | ○ | 274 |
| list-style-image | リストのマークの画像 | li要素 | none | ○ | 272 |
| list-style-position | リストのマークの表示位置 | li要素 | outside | ○ | 273 |
| list-style-type | リストのマークの形式 | li要素 | disc | ○ | 270 |
| margin | マージンの一括指定 | すべての要素 | 未定義 | × | 237, 256 |
| margin-top | 上マージン | すべての要素 | 0 | × | 237 |
| margin-bottom | 下マージン | すべての要素 | 0 | × | 237 |
| margin-left | 左マージン | すべての要素 | 0 | × | 237, 256 |
| margin-right | 右マージン | すべての要素 | 0 | × | 237, 256 |
| marker-offset | リストのマークと内容の間隔 | 「display: marker」の要素 | auto | × | |
| marks | トンボの指定 | @pageの宣言内のみ | none | - | |
| max-height | 最大の高さ | テキストのインライン要素とtable関連要素以外 | none | × | |

リファレンス

| プロパティ名 | 説明 | 適用対象 | 初期値 | 継承 | 参照 |
|---|---|---|---|---|---|
| max-width | 最大の幅 | テキストのインライン要素とtable関連要素以外 | none | × | |
| min-height | 最小の高さ | テキストのインライン要素とtable関連要素以外 | 0 | × | |
| min-width | 最小の幅 | テキストのインライン要素とtable関連要素以外 | ブラウザに依存 | × | |
| orphans | ページ最後の段落の必要行数 | ブロックレベル要素 | 2 | ○ | |
| outline | アウトラインの一括指定 | すべての要素 | 個別に参照 | × | |
| outline-color | アウトラインの色 | すべての要素 | invert | × | |
| outline-style | アウトラインの形式 | すべての要素 | none | × | |
| outline-width | アウトラインの太さ | すべての要素 | medium | × | |
| overflow | はみ出る部分の表示方法 | ブロックレベル・置換要素 | visible | × | 268 |
| padding | パディングの一括指定 | すべての要素※ | 未定義 | × | 239 |
| padding-top | 上パディング | すべての要素※ | 0 | × | 239 |
| padding-bottom | 下パディング | すべての要素※ | 0 | × | 239 |
| padding-left | 左パディング | すべての要素※ | 0 | × | 239 |
| padding-right | 右パディング | すべての要素※ | 0 | × | 239 |
| page | 用紙設定の適用 | ブロックレベル要素 | auto | ○ | |
| page-break-after | 要素の直後で改ページ | ブロックレベル要素 | auto | × | 281 |
| page-break-before | 要素の直前で改ページ | ブロックレベル要素 | auto | × | 281 |
| page-break-inside | 要素内での改ページの禁止 | ブロックレベル要素 | auto | ○ | |
| pause | 読み上げ休止時間の一括指定 | すべての要素 | ブラウザに依存 | × | |
| pause-after | 要素の直後の読み上げ休止時間 | すべての要素 | ブラウザに依存 | × | |
| pause-before | 要素の直前の読み上げ休止時間 | すべての要素 | ブラウザに依存 | × | |
| pitch | 読み上げ時の声の高さ | すべての要素 | medium | ○ | |
| pitch-range | 読み上げ時の声の高さの変化幅 | すべての要素 | 50 | ○ | |
| play-during | 読み上げ時のバックグラウンド音 | すべての要素 | auto | × | |
| position | 相対配置・絶対配置・固定配置 | CSSで追加される内容以外 | static | × | 258, 260, 262 |
| quotes | 引用符の設定 | すべての要素 | ブラウザに依存 | ○ | |
| richness | 読み上げ時の声の明瞭度 | すべての要素 | 50 | ○ | |
| right | 右からの位置 | 「position: static」以外の要素 | auto | × | 258, 260, 262 |
| size | 用紙サイズと方向 | @pageの宣言内のみ | auto | - | |
| speak | 音声読み上げ時の読み方 | すべての要素 | normal | ○ | |
| speak-header | 表の見出しの読み上げ方 | th要素・td要素 | once | ○ | |

※ただし、table関連要素で例外あり

| プロパティ名 | 説明 | 適用対象 | 初期値 | 継承 | 参照 |
|---|---|---|---|---|---|
| speak-numeral | 数字の読み上げ方 | すべての要素 | continuous | ○ | |
| speak-punctuation | 句読点や記号の読み上げ方 | すべての要素 | none | ○ | |
| speech-rate | 読み上げる速度 | すべての要素 | medium | ○ | |
| stress | 読み上げ時のアクセントの強さ | すべての要素 | 50 | ○ | |
| table-layout | テーブルの表示方法 | table要素 | auto | × | |
| text-align | 行揃え | ブロックレベル要素 | ブラウザと文字表記の方向に依存 | ○ | 214 |
| text-decoration | 下線・上線・取消線・点滅 | すべての要素 | none | × | 208 |
| text-indent | 1行目のインデント | ブロックレベル要素 | 0 | ○ | 219 |
| text-shadow | 影文字 | すべての要素 | none | × | |
| text-transform | 大文字・小文字 | すべての要素 | none | ○ | 222 |
| top | 上からの位置 | 「position: static」以外の要素 | auto | × | 258, 260, 262 |
| unicode-bidi | 文字方向の上書きと組み込み | すべての要素 | normal | × | |
| vertical-align | 縦方向の位置 | インライン要素・th要素・td要素 | baseline | × | 215 |
| visibility | 表示・非表示 | すべての要素 | visible | ○ | 266 |
| voice-family | 音声読み上げ時の声の種類 | すべての要素 | ブラウザに依存 | ○ | |
| volume | 音声読み上げ時の音量 | すべての要素 | medium | ○ | |
| white-space | 空白・改行・タブの処理 | すべての要素 | normal | ○ | 220, 221 |
| widows | ページ最初の段落の必要行数 | ブロックレベル要素 | 2 | ○ | |
| width | ボックスの内容領域の幅 | テキストのインライン要素とcol・colgroup要素以外 | auto | × | 249 |
| word-spacing | 単語間隔 | すべての要素 | normal | ○ | 217 |
| z-index | 重なる順序 | 「position: static」以外の要素 | auto | × | 264 |

# CSS2対応状況一覧

| プロパティ名 | Windows版 | | | | | | | | | | Macintosh版 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IE6 | IE5.5 | IE5.0 | IE4 | Firefox | N7 | N6 | N4 | O7 | O6 | Safari | IE5 | IE4 |
| background | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| background-attachment | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| background-color | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| background-image | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | △ | ○ | △ | △ |
| background-position | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | △ | ○ | ○ | ○ | △ |
| background-repeat | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| border | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| border-collapse | ○ | ○ | ○ | × | ○ | ○ | × | × | ○ | ○ | ○ | × | × |
| border-color | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| border-spacing | × | × | × | × | ○ | ○ | ○ | × | ○ | ○ | ○ | × | × |
| border-style | ○ | ○ | △ | △ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | △ |
| border-top | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| border-bottom | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| border-left | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| border-right | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| border-top-color | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| border-bottom-color | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| border-left-color | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| border-right-color | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| border-top-style | ○ | ○ | △ | △ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | △ |
| border-bottom-style | ○ | ○ | △ | △ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | △ |
| border-left-style | ○ | ○ | △ | △ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | △ |
| border-right-style | ○ | ○ | △ | △ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | △ |
| border-top-width | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| border-bottom-width | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| border-left-width | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| border-right-width | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| border-width | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| bottom | ○ | ○ | ○ | × | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | × |
| caption-side | × | × | × | × | ○ | ○ | △ | × | △ | △ | △ | × | × |
| clear | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| clip | × | × | × | × | × | × | × | × | × | × | × | × | × |

| プロパティ名 | Windows版 | | | | | | | | | | Macintosh版 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IE6 | IE5.5 | IE5.0 | IE4 | Firefox | N7 | N6 | N4 | O7 | O6 | Safari | IE5 | IE4 |
| color | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| content | × | × | × | × | △ | △ | △ | × | ○ | △ | △ | × | × |
| counter-increment | × | × | × | × | × | × | × | × | ○ | ○ | × | × | × |
| counter-reset | × | × | × | × | × | × | × | × | ○ | ○ | × | × | × |
| cursor | ○ | △ | △ | △ | ○ | ○ | ○ | × | ○ | × | ○ | ○ | △ |
| direction | ○ | ○ | ○ | × | ○ | ○ | ○ | × | ○ | × | × | × | × |
| display | △ | △ | △ | × | ○ | ○ | ○ | △ | ○ | ○ | ○ | △ | × |
| empty-cells | × | × | × | × | ○ | ○ | ○ | × | ○ | ○ | ○ | × | × |
| float | ○ | ○ | ○ | △ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | △ |
| font | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| font-family | ○ | ○ | ○ | △ | ○ | ○ | ○ | △ | ○ | ○ | △ | ○ | ○ |
| font-size | ○ | △ | △ | △ | ○ | ○ | ○ | △ | ○ | △ | ○ | ○ | △ |
| font-size-adjust | × | × | × | × | ○ | ○ | × | × | × | × | × | × | × |
| font-stretch | × | × | × | × | × | × | × | × | × | × | × | × | × |
| font-style | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| font-variant | ○ | △ | △ | △ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | △ |
| font-weight | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| height | ○ | △ | △ | △ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | △ |
| left | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| letter-spacing | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | △ | △ |
| line-height | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| list-style | △ | △ | △ | △ | ○ | ○ | ○ | △ | ○ | △ | ○ | △ | △ |
| list-style-image | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | △ | △ |
| list-style-position | ○ | ○ | ○ | △ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | × |
| list-style-type | △ | △ | △ | △ | ○ | ○ | ○ | △ | △ | △ | △ | △ | △ |
| margin | ○ | △ | △ | △ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | △ |
| margin-top | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| margin-bottom | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| margin-left | ○ | △ | △ | △ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | △ |
| margin-right | ○ | △ | △ | △ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | △ |
| marker-offset | × | × | × | × | × | × | × | × | × | × | × | × | × |
| marks | × | × | × | × | × | × | × | × | × | × | × | × | × |
| max-height | × | × | × | × | ○ | ○ | ○ | × | ○ | ○ | × | × | × |
| max-width | × | × | × | × | ○ | ○ | ○ | × | ○ | ○ | ○ | × | × |

| プロパティ名 | Windows版 | | | | | | | | | | Macintosh版 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IE6 | IE5.5 | IE5.0 | IE4 | Firefox | N7 | N6 | N4 | O7 | O6 | Safari | IE5 | IE4 |
| min-height | × | × | × | × | ○ | ○ | ○ | × | ○ | ○ | × | × | × |
| min-width | × | × | × | × | ○ | ○ | ○ | × | ○ | ○ | ○ | × | × |
| orphans | × | × | × | × | × | × | × | × | ○ | × | × | × | × |
| outline | × | × | × | × | × | × | × | × | ○ | × | △ | ○ | × |
| outline-color | × | × | × | × | × | × | × | × | ○ | × | △ | ○ | × |
| outline-style | × | × | × | × | × | × | × | × | ○ | × | △ | ○ | × |
| outline-width | × | × | × | × | × | × | × | × | ○ | × | △ | ○ | × |
| overflow | △ | △ | △ | × | ○ | ○ | ○ | × | ○ | △ | ○ | △ | × |
| padding | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| padding-top | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| padding-bottom | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| padding-left | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| padding-right | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| page | × | × | × | × | × | × | × | × | × | × | × | × | × |
| page-break-after | ○ | ○ | ○ | △ | ○ | ○ | × | × | ○ | ○ | ○ | △ | △ |
| page-break-before | ○ | ○ | ○ | △ | ○ | ○ | × | × | ○ | ○ | ○ | ○ | △ |
| page-break-inside | × | × | × | × | × | × | × | × | ○ | ○ | × | × | × |
| position | △ | △ | △ | △ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | △ |
| quotes | × | × | × | × | △ | △ | △ | × | ○ | ○ | × | × | × |
| right | ○ | ○ | ○ | × | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | × |
| size | × | × | × | × | × | × | × | × | ○ | ○ | × | × | × |
| table-layout | ○ | ○ | ○ | × | ○ | ○ | ○ | × | ○ | ○ | ○ | × | × |
| text-align | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | △ |
| text-decoration | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | △ |
| text-indent | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| text-shadow | × | × | × | × | × | × | × | × | × | × | ○ | × | × |
| text-transform | ○ | ○ | ○ | ○ | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| top | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| unicode-bidi | ○ | ○ | ○ | × | ○ | ○ | ○ | × | ○ | × | × | × | × |
| vertical-align | ○ | ○ | △ | △ | ○ | ○ | ○ | △ | ○ | ○ | ○ | △ | △ |
| visibility | △ | △ | △ | △ | ○ | ○ | ○ | × | △ | △ | △ | △ | △ |
| white-space | ○ | △ | × | × | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | × |
| widows | × | × | × | × | × | × | × | × | ○ | × | × | × | × |

| プロパティ名 | Windows版 | | | | | | | | | | Macintosh版 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | IE6 | IE5.5 | IE5.0 | IE4 | Firefox | N7 | N6 | N4 | O7 | O6 | Safari | IE5 | IE4 |
| width | ○ | △ | △ | △ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |
| word-spacing | ○ | × | × | × | ○ | ○ | ○ | × | ○ | ○ | ○ | ○ | ○ |
| z-index | ○ | ○ | ○ | ○ | ○ | ○ | ○ | △ | ○ | ○ | ○ | ○ | ○ |

※各プロパティの対応状況を、○△×の３段階でおおまかに示しました。
環境や細かいバージョンの違いなどによっては、結果が異なる場合もありますので、注意してください。
標準仕様とは異なる指定方法が必要なものについては、未対応としています。
※音声ブラウザ用などの一部のプロパティは、省略しています。

**注意**

## ソースをコピーする時の注意

JavaScriptは、ソースコードを直接HTMLファイルに書き込むため、比較的簡単にソースコードを見ることができます。これは、インターネット上に数多く公開されているコピーフリーのJavaScriptのソースを簡単に見ることができるということなので、JavaScriptを勉強する上で非常に有意義なことであり、そしてこれはJavaScriptの大きな魅力のひとつでもあります。

しかし、Netscape Navigator 3.0以上のバージョンのブラウザで、[表示]メニューの[ページのソース]を使ってソースコードを見ようとした場合、そこに表示されるソースコードは、正確ではない場合があります。

たとえば、次のスクリプトを実行した時、

```
<script language="JavaScript">
document.write("今日は!!")
</script>
```

[ページのソース]で表示されるソースは、JavaScriptのコード部分がなくなり、次のようになる場合があります。

```
今日は!!
```

また、これと同じような例で、特殊フォントの「<」や「≶」も「<」や「>」となります。

このように、[ページのソース]で表示されるソースは、HTMLファイルの内容そのままではなく、一旦ブラウザに読み込まれて評価された結果のHTMLになることがあるのです。

この問題を回避してJavaScriptのソースを正確に見るには、ブラウザの設定でJavaScriptの実行を一旦中止してからページを読み込み、ページのソースを表示するようにすれば大丈夫です。

詳解

HTML&CSS&JavaScript辞典 改訂版

The revised edition of the detailed explanation dictionary about HyperText Markup Language, Cascading Style Sheets, and JavaScript.

# JavaScript

# JavaScriptとは？

## JavaScriptとは

JavaScriptとは、Netscape社がWebページの処理能力を高めるために開発したLiveScriptを元に、Netscape社とSun社が共同で開発したスクリプト言語で、Netscape Navigator 2.0以降のブラウザとInternet Explorer 3.0以降のブラウザで対応されています。

JavaScriptを使うことにより、Webページを動的に変化させたり、今までCGIなどで行う必要があった処理の一部を、Webページ上で行うことが可能になります。

仕様にJavaと似た部分があり、JavaScriptが実行できる環境(ブラウザ)さえあればOSが違っていても同じように動く(ことを期待できる)プログラムを書くことができる、などの点がJavaと似ているといえます。けれども、基本的にJavaとは別物と考えたほうがよいでしょう。

Javaとの最大の違いは、コンパイルをする必要がなく、HTML文章内に直接JavaScriptを記述し、そのファイルをブラウザで読み込むことによって、手軽にスクリプトを実行できる点が挙げられます。

## JavaScriptの種類

JavaScriptは、現在JavaScript1.5までのバージョンが公開されています。JavaScriptのより新しいバージョンは、一部の変更点を除いて、古いバージョンのJavaScriptのすべてに対応しています。

それぞれのバージョンの特徴と対応ブラウザは、次の通りです。

### ■JavaScript1.0

ブラウザのウィンドウを操作するwindowオブジェクトや、日付を取り扱うDateオブジェクトなどの基本的なオブジェクトが追加されました。

Internet Explorer 3.0、Netscape Navigator 2.0以降のブラウザでサポートされています。

### ■JavaScript1.1

画像を取り扱うImageオブジェクトがサポートされました。これによりページ表示後に画像が置き換えられるようになり、マウスポインタの動作によって画像を差し換えたり、定期的に画像を差し換えることによってアニメーションの効果を出したりできるようになりました。

Internet Explorer 4.0、Netscape Navigator 3.0以降のブラウザでサポートされています。

### ■JavaScript1.2

ディスプレイサイズなどのディスプレイの情報を取得するscreenオブジェクトや、ウィンドウ上に絶対座標でコンテンツの位置を設定したり、重なりを指定することができるLayerオブジェクトなどが追加されました。これにより、ディスプレイ上のウィンドウの位置や、ウィンドウ内に表示するコンテンツの位置や重なりを細かく設定す

ることができるようになり、さらにそれらを動的に変化させることができるようになりました。

Internet Explorer 4.0、Netscape Navigator 4.0以降のブラウザでサポートされています。

## ■JavaScript1.3

文字コードがの扱いがUnicodeになったほか、日付を取り扱うDateオブジェクトで年号が4桁で表せるようになったり、ミリセコンド単位の時間を扱えるようになったり、といった細かな部分で追加·変更が行われています。これらの処置は、ECMAScript(ECMA-262)の1版と互換をとるためのものです。

Internet Explorer 5.0、Netscape Navigator 4.06以降のブラウザでサポートされています。

## ■JavaScript1.4

ECMAScriptの1版と完全互換したスクリプトです。

JavaScript1.4を搭載したブラウザは、結局は発表されませんでした。

## ■JavaScript1.5

ECMAScriptの第3版と完全互換を持ったスクリプトです。

標準規格に完全に準拠する姿勢に合わせてJavaScript1.5として規格化されているのは、ECMAScriptと同様にビルトインオブジェクトの部分のみです。しかし、DOM(Document Object Model)のコントロールを行う方法は規定されているので、DOMでサポートされている、ブラウザやHTML、スタイルシートなどのあらゆる要素を、JavaScriptを使って操作することができます。

Mozillaや Netscape 6.0以降のブラウザでサポートされています。また、Internet Explorerは、バージョン5.0以降であれば、JavaScript1.5レベルのスクリプトをサポートしています。

この他のJavaScriptと共に覚えておきたいスクリプト言語に、ECMAScriptとJScriptがあります。これらはJavaScriptをベースにして作られた言語で、JavaScriptと互換をとるように配慮されていますが、その使用や実装には微妙な違いがあり、残念ながら現状では100%互換とは言い切れません。

それぞれの言語の特徴は、次の通りです。

## ■ECMAScript

ヨーロッパの標準化機関であるECMA(European Computer Manufacturers Association)が、JavaScript1.1をベースに規定したインターネットで使用するスクリプト言語の仕様です。ECMA-262として仕様が公開されています。

ECMAScriptでは、JavaScriptのビルトインオブジェクトの部分は規格化されていますが、ナビゲータオブジェクトの部分でJavaScriptのように細かくオブジェクトを規格化するようなことは行われていません。しかし、オブジェクトの取り扱い方は規定されているので、これを利用してDOMをコントロールすることが可能です。

ECMAScriptはあくまでも言語仕様を規定したものであり、その仕様に合わせてどの

ように実装されるかは、その言語によって変わってきます。現在、JavaScript1.3以降とMicrosoft社のJScript3.1以降が、ECMAScriptと互換がとられていることになっていますが、ふたつのスクリプト言語にはかなり違う部分があります。また、SafariとOperaは、正確にはこのECMAScriptをサポートしたブラウザとなります。このため、両ブラウザは、JavaScript1.5レベルのJavaScriptを、ほぼサポートしています。

ECMA-262は、現在3版まで公開されています。JavaScriptとECMA-262の各版との関連は、次の通りです。

| バージョン | 関連性 |
| --- | --- |
| JavaScript1.1 | これを元にECMA-262の1版が作成された |
| JavaScript1.2 | ECMA-262では、Unicodeの採用による国際化が行なわれているほか、JavaScriptにおけるDateオブジェクトのtoGMTStringメソッドのように、実行結果がマシン環境に左右されることのないような規格が作られている。また、JavaScript1.2では、ECMA-262の1版では考慮されていなかった独自の仕様が追加されているため、完全な互換性は持っていない |
| JavaScript1.3 | ECMA-262の1版と完全互換を持つように変更が行われた |
| JavaScript1.4 | ECMA-262の1版と完全互換を持つ(JavaScript1.4を採用したブラウザは存在しない) |
| JavaScript1.5 | ECMA-262の3版と完全互換を持つ |

なお2版は、1版の細かい部分の変更やバグフィックスからなります。ECMA-262の4版は、現在ECMAのTC39ワーキンググループで話し合われています。これは、将来JavaScript2.0になる予定です。

## ■ JScript

Microsoft社が独自に開発したJavaScript互換のスクリプト言語です。Internet Explorerに搭載されているのは、正確にいうとJavaScriptではなく、このJScriptということになります。

Internet Explorer 3.0(Mac版はInternet Explorer 3.1)から実装されており、Internet Explorer 3.XにはJavaScript1.0レベルのスクリプトが実行できるJScriptが、Internet Explorer 4.XではJavaScript1.1レベルのスクリプトとJavaScript1.2の一部が実行できるレベルのJScriptが実装されています。

現在仕様が公開されているJScriptはJScript5.0で、これはJavaScript互換というよりもECMAScript互換スクリプトといった方がよく、ECMAScriptに準拠しながらInternet Explorer独自の拡張が施されていて、Internet Explorer 5.X以降で採用されています。

## ブラウザから見たJavaScriptの対応状況

Internet ExplorerとNetscape Navigator両ブラウザの各バージョンで対応しているJavaScriptと、JavaScriptを使う上での注意点は、次の通りです。

### ■Internet Explorer 3.X

JavaScript1.0レベルのスクリプトに対応しています。

Internet Explorer 3.Xは、たとえ同じバージョン番号のものであっても細かい改良が加えられており、実行できるJavaScriptも増えてきています。その結果、同じバージョンのInternet Explorerで同じスクリプトを実行したとしても、一方のブラウザは正常に実行でき、一方のブラウザはエラーになる、という現象が発生します。

Internet Explorer 3.Xは、JavaScript以外でも、HTMLの表示やセキュリティの問題など多くの部分が改良されているので、なるべく最新のものを使用することをお勧めします。

### ■Internet Explorer 4.X

JavaScript1.1レベルのスクリプトに対応しています。

JavaScript1.2レベルのスクリプトも、レイヤーなどのNetscape Navigator独自の部分を除いて、ほぼ対応しています。ECMAScript（ECMA-262）の仕様に準拠しているので、正式にはJavaScript1.3には対応していませんが、JavaScript1.3の多くのスクリプトが実行可能です。

また、DOMもこのバージョンからサポートされており、スタイルシート関連の要素を始め、Internet Explorerでサポートされているほとんどの要素を、オブジェクトとして取り扱うことが可能です。しかし、その実装は、DOMの規格が確定する前だったこともあり、今となってはInternet Explorerの独自色が強いものとなっています。

### ■Internet Explorer 5.X/6.0

JavaScriptのサポートに関しては、Internet Explorer 4.Xとほぼ同等です。JavaScript1.3にも対応しています。

DOMの扱いも、規格に合わせた実装が目指されているため、多くの場合、Netscape Navigator 6.XやMozillaと同じ用法で使用することができます。

### ■Netscape Navigator 2.X

JavaScript1.0に対応しています。

JavaScriptに始めて対応したブラウザです。Macintosh版では、新しいウィンドウを開いた時に表示するページのURLが取得できずにページが表示できない、ループしてJavaScriptを実行するとメモリーエラーが発生する、などの問題がありました。

### ■Netscape Navigator 3.X

JavaScript1.1に対応しています。

Netscape Navigator 2.XのMacintosh版にあった問題点が解消されています。

## ■Netscape Navigator 4.X

JavaScript1.2に対応しています。

レイヤーなど、Netscape Navigator独自仕様のものもJavaScriptで制御できるようになりましたが、その結果、スクリプトの独自色が強まってしまいました。

また、W3Cによる標準仕様が存在するスタイルシートもレイヤーと同じようにJavaScriptで制御できますが、Netscape Navigator自身のスタイルシートの実装には、不完全な部分があります。

## ■Netscape Navigator 4.06以降

JavaScript1.3に対応しています。

文字がUnicodeで扱われるようになったため、ページ上やフォーム、ステータス行などにJavaScriptを使用して文字を書き出す場合に、表示される文字が文字化けを起こす可能性があります(この問題はNetscape Navigator 4.5日本語版では解消されています)。また、Unicodeでは文字が2バイト文字なので、文字数を数えているようなスクリプトは、今までとは違う結果になる場合があります。

## ■Netscape 6.X / 7.X / Mozilla / Firefox

JavaScript1.5に対応しています。

標準規格であるECMAScriptと完全互換を持つほか、DOMのサポートにより、今まで以上に細かい部分までブラウザを制御することが可能になりました。



**注意**

### Netscape NavigatorとInternet ExplorerとECMAScriptの関係

Netscape NavigatorでのECMAScriptのサポートはJavaScript1.3から、つまりNetscape Navigator 4.06からです。それに対してInternet Explorerは、Internet Explorer 4.XからECMAScriptをサポートしています。また、Netscape Navigator 4.06以前のNetscape Navigator 4.Xのブラウザでも、一部ECMAScriptをサポートしています。

つまり、Internet Explorer 4.XやNetscape Navigator 4.06以前のNetscape Navigator 4.Xは、<script>内の「language」属性の設定で、「<script language="JavaScript1.3">」と設定されたスクリプトは実行できませんが、「getFullYear()」メソッドなどのJavaScript1.3に含まれる多くのECMAScriptを実行することができます。

# JavaScriptの記述法

JavaScriptは、HTMLファイル内にHTMLタグを使ってJavaScriptであることを指定し、そのタグ内にソースコードを記述することによって設定します。

JavaScript関連で使用するタグと、それらのタグを使ってHTMLファイル内へのJavaScriptを記述する方法は、次の通りです。

## <script>の使い方

HTML内にJavaScriptを記述するには、<script>を使用します。

<script>内で「language」属性を指定する時に、「language="JavaScript"」と記述しておくと、ブラウザがその中に記述されている文字列はJavaScriptであると判断して実行します。

Imageオブジェクトなどの JavaScript1.1を使ったスクリプトを記述する場合には、「language」属性の指定を「language="JavaScript1.1"」とします。こうやって指定された<script>内のスクリプトは、Netscape Navigator 3.0以上などのJavaScript1.1に対応したブラウザでのみ実行され、Netscape Navigator 2.0などのJavaScript1.1未対応のブラウザでは実行されません。

また、Layerオブジェクトなどの JavaScript1.2を使ったスクリプトを記述する時は、「language」属性の指定を「language="JavaScript1.2"」とします。こうやって指定された<script>内のスクリプトは、Netscape Navigator 4.0などのJavaScript1.1に対応したブラウザでのみ実行されるようになります。

同様に、JavaScript1.3を使ったスクリプトを記述する場合の「language」属性の指定は「language="JavaScript1.3"」となります。こうやって指定された<script>内のスクリプトは、Netscape Navigator 4.5などのJavaScript1.3に対応したブラウザでのみ実行されます。

このように、あらかじめ「language」属性でJavaScriptのバージョンを記述しておけば、記述したバージョンに対応していないブラウザではそのスクリプトは実行されません。これによりJavaScriptに未対応のブラウザで見た時に起こるエラーを回避することができます。

ブラウザは、基本的に自分の知らないタグを無視します。そのため、JavaScriptに未対応のブラウザでJavaScriptが記述しているページを見ると、ソース部分が丸見えになってしまいます。

これを防ぐため、スクリプトの1行目を「<!--」に、最終行を「//-->」とすることによって、ソースをコメントアウトします。JavaScript未対応のブラウザで閲覧している人に迷惑をかけることがないように、「おまじない」とでも考えて必ず記述するようにしてください。

具体的な<script>の書き方は、次のようになります。

この「JavaScript1.Xのソース」の部分に実際のスクリプトを記述します。スクリプトは、HTMLファイルに記述されている上から順に評価され、より下に記述されているものが優先されます。

### ■JavaScript1.0の記述法

```
<script language="JavaScript">
<!--
JavaScript1.0のソース
//-->
</script>
```

### ■JavaScript1.1の記述法

```
<script language="JavaScript1.1">
<!--
JavaScript1.1のソース
//-->
</script>
```

### ■JavaScript1.2の記述法

```
<script language="JavaScript1.2">
<!--
JavaScript1.2のソース
//-->
</script>
```

### ■JavaScript1.3の記述法

```
<script language="JavaScript1.3">
<!--
JavaScript1.3のソース
//-->
</script>
```

## MozillaやNetscape 6.X以降での<script>の書き方

Netscape 6.0以降と、その元となったMozillaでは、ECMAScriptをはじめ、HTML4.0・CSS・XML・DOMなどの多くの標準化された仕様の採用と、それの100%互換を目指しています。そして、これに従って<script>の記述方法にも変化があります。具体的には、HTML4.0では「language」属性は不適切となり、スクリプトの指定には「type」属性を使うようになっています。そのため、MozillaやNetscape 6.Xでの<script>の書き方は、「type」属性を使い、次のようになります。

```
<script type="text/javascript">
<!--
JavaScriptのソース
//-->
</script>
```

また、HTML4.0の仕様に従うのなら、<head>～</head>内の最初で<meta>を使って、スクリプト言語を指定する必要もあります。

この場合の<meta>は、次のような設定になります。

```
<meta http-equiv="Content-Script-Type" content="text/javascript">
```

しかし、この記述方法では、スクリプトのバージョン指定がうまく行えないという問題が発生します。たとえば、「type」属性内で「<script type="text/javascript; version=1.2">」とバージョン指定を行うと、Internet Explorer 4.X ～ 5.XやNetscape Navigator 4.Xなど、本来JavaScript1.2が実行できるブラウザでもスクリプトは実行されません。

未サポートのバージョンのJavaScriptから来る、不必要なエラーを回避するためには、スクリプトのバージョン指定は大変に有効な手段です。しかしながら今後は、ここで紹介した、よりHTMLの仕様に沿った「language」属性を使わない記述方法が、主流になっていくでしょう。

## 本書での「script」タグ要素の記述について

HTML4.0以降、「script」要素の「language」属性が不適切とされるようになりました。このことは、JavaScriptを指定するタグの書き方に関しては、大きな変更と言えます。確かに「language」属性を設定することで未対応のバージョンのJavaScriptを実行できないようにすることは、手軽でとても有効な手段です。しかし、Internet Explorerに加え、Netscape 6.Xをはじめとして、HTML4.01に対応したブラウザも増えてきました。また、今後発表されてくる新しいブラウザも、HTML4.01かそれ以降のバージョン

### 注意

### 「language」属性の記述について

JavaScriptを記述する時に「language」属性を省略して次のように記述しても、Internet Explorer、Netscape Navigatorどちらのブラウザも、<script>内に記述されたスクリプトをJavaScriptとして解釈して、実行します。

```
<script>
<!---
JavaScriptのソース
//-->
</script>
```

HTML4.0では、「language」属性は不適切となりました。しかし、「language」属性を設定してJavaScriptのバージョンを記述することにより、そのバージョンに対応していないブラウザで発生するエラーが簡単に回避できます。このことからも、HTML4.0に完全に準拠したHTMLを記述する場合以外でも、「language」属性はきちんと設定しておくべきでしょう。

のHTMLをサポートすることになるであろうことを考えると、現段階では、HTML4.01に準拠したHTMLを記述するようにするべきでしょう。

その考えから、本書では、body要素内に設定するブラウザの背景色や文字色などの値を変更するような一部のスクリプトを除き、HTML4.0以降のHTMLに準じることにします。具体的には、<meta>で「<meta http-equiv="Content-Script-Type" content="text/javascript">」としてJavaScriptを使用していることを宣言し、<script>も「<script type="text/javascript">」を使用しています。

## <noscript>の使い方

<noscript>は、JavaScriptを無効にしているか、対応していないブラウザを使っているユーザーに対してメッセージを表示する時に使用します。

<noscript>は、<script>～</script>外で使用し、JavaScriptが有効な時には<noscript>～</noscript>内の記述は表示されません。

具体的な<noscript>の書き方は、次のようになります。

```
<noscript>
このページではJavaScriptが使われています...。
</noscript>
```

## コメントの書き方

JavaScript内でのコメントを書くには、次の2通りの方法があります。

「//」では以降の1行が、「/*～*/」では間に挟まれた文字列が、コメントとして扱われます。

ソースコードをコメントアウトする時に最後の行に使用する「//-->」の「//」の部分もこれに当たります。

具体的なコメントの書き方は、次のようになります。

```
<script type="text/javascript">
<!--
//この1行はコメントとなります。
/*これに囲まれている部分
  はコメントとなります。*/
//-->
</script>
```

## 文の区切り方

「;」は文の区切りを表します。JavaScriptでは、たとえ改行があったとしても、「;」によって文を区切っていない限り、ひとつの文として扱われます。

JavaScriptは、基本的に「;」を省略しても自動的に文の区切りが判断されて、スクリプトが正常に処理されますが、ソースをわかりやすくする意味も含めて、文は「;」で区切っておくことをお勧めします。

## JavaScriptの外部呼び込み方法

JavaScriptは、HTMLファイル内に直接記述する方法以外に、外部にスクリプトを記述したファイルを置き、<script>内で「src」属性を使用してURLを指定することにより、それを読み込んで実行することが可能です。指定したファイル内には、前後の<script>を省略したJavaScriptのソースコードを記述します。この時のファイル名の拡張子は「.js」とします。

```
<script src="URL" type="text/javascript"></script>
```

この方法は、スクリプトのソースコードを見えにくくしたり(完全に隠蔽することはできません)、複数のページで同一のスクリプトを使用する場合は特に有効です。たとえば、ファイルの更新日時を複数のWebページ上に表示するような場合は、必要な全ページにひとつひとつスクリプトを記述するのではなく、次のようにファイルの最終更新日時を書き出すスクリプトを書いて、適当な名前を付けた拡張子「.js」のファイルをひとつ作ります。後は、全ページにそのファイルのURLを指定した<script>を、表示したい位置に記述するだけで済みます。

```
document.write("Last update:",document.lastModified)
```

この方法は、Internet Explorer 4.0以降、Netscape Navigator 3.0以降から使用可能です。

また、本来ならば、<script>と同様に「language」属性を使用してJavaScriptのバージョンを記述することによって、そのJavaScriptのバージョンに対応がとられた「.js」ファイルしか読み込まないようにできるはずなのですが、現在その機能は正常に動作しません。

「.js」ファイルを置くWebサーバーには、MIMEタイプの設定をしておく必要があります。

**注意**

### MIMEタイプの設定について

JavaScriptの外部読み込みをするためには、あらかじめサーバーにMIMEタイプの設定がされてなければいけません。

たとえば、WebサーバーにNCSAhttpdやApacheが使われていて、ユーザーのホームディレクトリで「.htaccess」ファイルによる設定が許されているような場合は、その中に次のように記述することによって設定します。

```
AddType application/x-javascript .js
```

MIMEタイプの設定には、この他にも色々な方法があり、「.htaccess」ファイルによる設定を行わなくても、プロバイダによっては初めからMIMEタイプが設定されている場合もあります。

MIMEタイプの設定に関しては、Webサーバーの運用に関わってくる問題も含まれるので、サーバー管理者の方と相談するようにしてください。

# オブジェクト・プロパティ・メソッド

## オブジェクト

JavaScriptでは、ブラウザの各部品や情報をオブジェクトとして取り扱うことができます。そして、このオブジェクトの値を変更したり、値を調べてそれによって違った処理を設定することによって、ブラウザを動的に変更することができるのです。

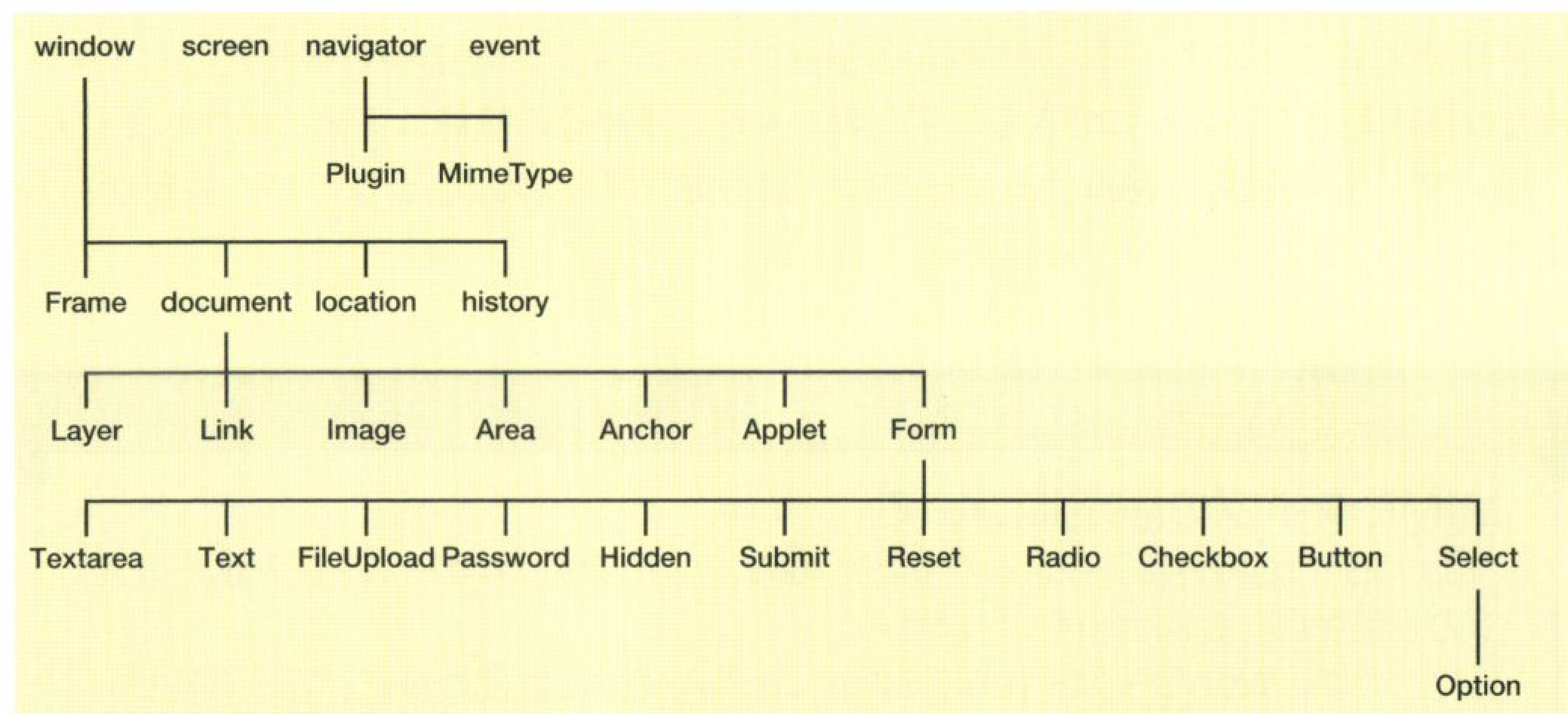
JavaScriptのオブジェクトは大きく分けると、ブラウザ自身が本来持っている部品や情報を取り扱うナビゲータオブジェクトと、独自に組み込まれたビルトイン(組み込み)オブジェクトの2種類があります。

### ■ナビゲータオブジェクト

ブラウザ自体の名前やバージョンといった情報や、ブラウザに表示されるドキュメント、画像、フォームなど、ブラウザがあらかじめ持っている部品を取り扱うオブジェクトのことを、ナビゲータオブジェクトといいます。

ナビゲータオブジェクトには階層関係があり、使用する時はその階層関の上から順番に「.」で区切って記述します。たとえばdocumentオブジェクトは、windowオブジェクトのひとつ下の階層にあるオブジェクトなので、「window.document」として表します。ただし、1番上の階層になるwindowオブジェクトは、省略することができます。

ナビゲータオブジェクトの階層関係は、下図の通りです。




ナビゲータオブジェクトの階層

### ■ビルトインオブジェクト

ブラウザ自信が本来持っているオブジェクトの他に、JavaScriptがブラウザに独自に組み込んでいるオブジェクトがあり、それをビルトインオブジェクトといいます。

JavaScriptでは、日付や時間などの時を取り扱うDateオブジェクトや、文字列の操作を行うstringオブジェクトなど、多くのビルトインオブジェクトが用意されています。

ユーザー独自のオブジェクトを作成したり、ビルトインオブジェクトを使用する時

は、new演算子を使います。new演算子によるオブジェクトの作成(インスタンスの作成)は、次の書式で行います。そしてそれ以降は、「オブジェクト名」で指定した名前を使って、オブジェクトの操作を行うことができるようになります。

```
オブジェクト名=new オブジェクトの型(値)
```

このオブジェクト名は、一定の条件下において、ユーザー側で自由に設定することができます。

## プロパティ

オブジェクトは多くの属性を持っており、この属性のことをプロパティといいます。

プロパティもまた、オブジェクトです。たとえばdocumentオブジェクトは、それ自体がオブジェクトであるのと同時に、windowオブジェクトのプロパティであるともいえます。

JavaScriptではプロパティはオブジェクトの後に「.」で区切って設定します。

```
オブジェクト.プロパティ
```

また、プロパティの中には、ユーザーが値を設定することができるものがあります。そのようなプロパティに値を設定する場合は、次の書式で設定します。

```
オブジェクト.プロパティ=値
```

各オブジェクトがどのようなプロパティを持っているかは、リファレンスの「ナビゲータオブジェクト」(P.625)・「ビルトインオブジェクト」(P.642)を参照してください。

## メソッド

メソッドは、オブジェクトに対して動作を指定します。

メソッドは、次のようにオブジェクトの後に「.」で区切って設定します。また、オブジェクトに値を設定する時には、「()」内に値を設定することによって行います。

```
オブジェクト. メソッド(値)
```

JavaScriptの各オブジェクトがどのようなメソッドを持っているかは、リファレンスの「ナビゲータオブジェクト」(P.625)・「ビルトインオブジェクト」(P.642)を参照してください。

## イベントハンドラ

ユーザーやスクリプトによってページがロードされたり、オブジェクトがクリックされたりというような、特定の動作が起こったタイミングをイベントといいます。JavaScriptでは、イベントの発生を取得して、そのタイミングでスクリプトの実行を開始することができます。

このイベントの取得を行うものを、イベントハンドラといいます。

イベントハンドラの設定は、そのイベントハンドラが設定可能なオブジェクトのHTMLタグ内に、次のような書式で設定します。

```
イベントハンドラ名=スクリプトまたは関数
```

JavaScriptで用意されているイベントハンドラと、そのイベントハンドラがどのようなイベントを取得し、どのオブジェクトに対応しているかは、リファレンスの「イベントハンドラ」(P.619)を参照してください。

## eventオブジェクトによるイベントの取得

JavaScript1.2からは、イベントをオブジェクトとして捕らえるeventオブジェクトが追加されました。これにより、イベントを取得したいオブジェクトに対して取得するイベントを設定することにより、そのオブジェクト上のどこからでもイベントの発生を取得することができるようになりました。また、取得したイベントからは、そのイベントのイベントタイプなど、イベントに関する色々な値を取得することが可能です。

イベントの取得は、次の用法で設定します。

```
オブジェクト.イベント=関数名またはスクリプト
```

JavaScriptで設定できるイベントタイプと各イベントが設定できるオブジェクト、イベントで取得できる値、つまりeventオブジェクトのプロパティにどのようなものがあるかは、リファレンスの「イベントタイプ」(P.622)を参照してください。

## JavaScriptで取り扱える型の種類

プロパティやメソッドに設定する値はもちろん、スクリプトが返す値や変数･定数など、JavaScriptで取り扱う値(データ)は、必ず何らかのデータ型を持っています。

JavaScriptで取り扱える型にどのようなものがあるかは、リファレンスの「JavaScriptで取り扱える型の種類」(P.608)を参照してください。

# 関数

一連の処理手続をまとめて名前を付けたものを、関数といいます。

JavaScriptでの関数は、大きく分けて関数の処理を定義する部分と、関数を呼び出す部分のふたつの部分からできています。

関数の処理の定義は、次のように記述します。

```
function 関数名(引数,引数,...){ 処理 }
```

この関数名は、一定の条件下において、ユーザー側で自由に設定することができます。

関数の呼び出しは、ページ上やイベントハンドラ内で行い、関数の処理を呼び出したい部分に、「関数名(引数,引数,...)」と記述して設定します。こうしておけば、ページが読み込まれた時や、特定のイベントが発生した時に関数の処理が呼び出され、定義した関数の処理が実行されます。

引数とは、その名の通り関数の処理に引き渡す値のことをいいます。関数の処理に値を引き渡す必要がない時は、引数を設定する必要はありません。

通常、関数の処理の定義は<head>内で行い、関数の処理の呼び出しは<body>内で行います。これは、関数の定義が終わらないうちに関数が呼び出されることを防ぐためです。

# ビルトイン関数

JavaScriptには、始めから定義されている関数があり、これをビルトイン関数といいます。

ビルトイン関数は、オブジェクトに依存することなく、スクリプト内のどこからでも使用することができます。

JavaScriptで用意されているビルトイン関数にどのようなものがあるかは、リファレンスの「ビルトイン関数(top-level関数)」(P.649)を参照してください。

## 変数・定数

変数とは、値を入れておく箱のようなものと考えればいいでしょう。文字列や数値、オブジェクト・メソッド・プロパティ、あるいは条件式などで設定する式や変数などを、変数に設定することができます。

変数は、次のようにして設定します。

```
var 変数名 = 値
```

変数名は、一定の条件下において、ユーザー側で自由に設定することができます。また、「var」は省略可能です。

このようにして設定すると、右辺の「変数名」に「値」が代入され、以降「変数名」を使用することによって、値を取り出すことができます。

さらに、後から同じ「変数名」に値を設定し直すことによって、変数の値を自由に変化させることもできます。

このように、値を設定し直すことによって絶えず値が変化する変数に対し、変数に設定する値の方は変化することはありません。このことから、変数に代入する値のことを定数(リテラル)といいます。

## オブジェクト・関数・変数などに設定可能な名前

オブジェクト名・関数名・引数名・変数名・定数名は、以下の条件下において、ユーザー側で自由に名前を定義することができます。

1. 大文字・小文字のアルファベット、あるいはアンダースコア"_"で始まる文字列(hamba,HAMBA,_hamba など)
2. 日本語(2バイト文字)は使用できない
3. スペース・コンマ・疑問符・引用符は使用できない
4. 文字列内に数値を入れることは可能だが、数値を先頭にすることはできない("Hamba1"や"f_3hamba"は可能だが、"6hamba"は不可)
5. 大文字小文字は区別される("HAMBA"と"hamba"は区別される)
6. 予約語は使用できないが、以上の条件を満たし、予約語を含む文字列("default"は不可だが、"Setdefault"は可)

### 予約語

システム側であらかじめ予約されている文字を予約語といい、オブジェクト名･関数名･引数名･変数名･定数名として使用することはできません。

予約語には、次のようなものがあります。

| | | | | |
|---|---|---|---|---|
| abstract | boolean | break | byte | case |
| catch | char | class | const | continue |
| default | delete | do | double | else |
| extends | false | final | finally | float |
| for | function | goto | if | implements |
| import | in | instanceof | int | interface |
| long | native | new | null | package |
| private | protected | public | return | short |
| static | super | switch | synchronized | this |
| throw | throws | transient | true | try |
| typeof | var | void | while | with |

## 演算子

値の計算や、比較などに用いる記号のことを演算子といいます。

JavaScriptで使用できる演算子にどのようなものがあるかは、リファレンスの「演算子」(P.609)を参照してください。

## JavaScriptの命令文（ステートメント）

JavaScriptでは、if文やfor文をはじめ、多くの命令分を使うことができます。

JavaScriptで使うことができる命令文と、その使い方については、リファレンスの「JavaScriptの命令文（ステートメント）」(P.613)を参照してください。

JavaScript

# ブラウザ名を取得する

## navigator.appName ［プロパティ］

Internet Explorer 6.0




Firefox 1.0




「appName」プロパティは、ブラウザ名の値を持っています。
Internet ExplorerやOperaは「Microsoft Internet Explorer」を、Netscape NavigatorやMozilla、Firefox、Safariは「Netscape」を返します。
このプロパティは読み出し専用です。

### Sample

```
<script type="text/javascript">
<!--
document.write("ブラウザ名:",navigator.appName);
//-->
</script>
```

JavaScript

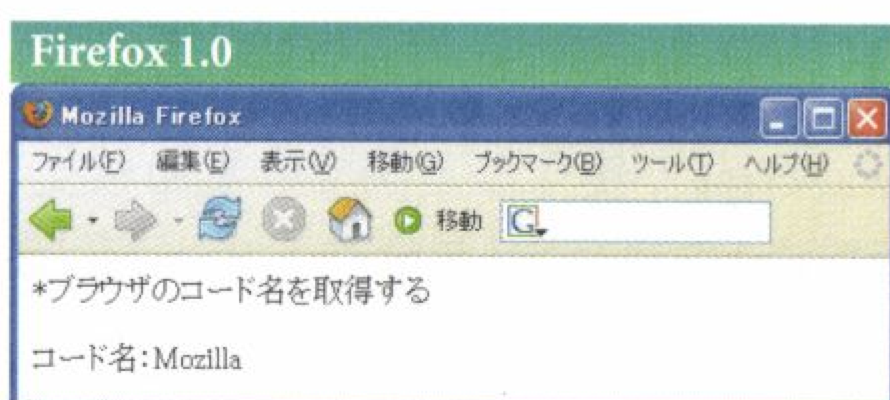
# ブラウザのコード名を取得する

## navigator.appCodeName ［プロパティ］

Internet Explorer 6.0




Firefox 1.0




「appCodeName」プロパティは、ブラウザのコード名の値を持っています。
Netscape NavigatorやMozilla、Firefox、Safari、Operaは「Mozilla」を、Internet Explorerは「Mozilla/3.0 (compatible; MSIE 3.01; Mac_PowerPC)」や「MSIE」「Mozilla」などを返します。
このプロパティは読み出し専用です。

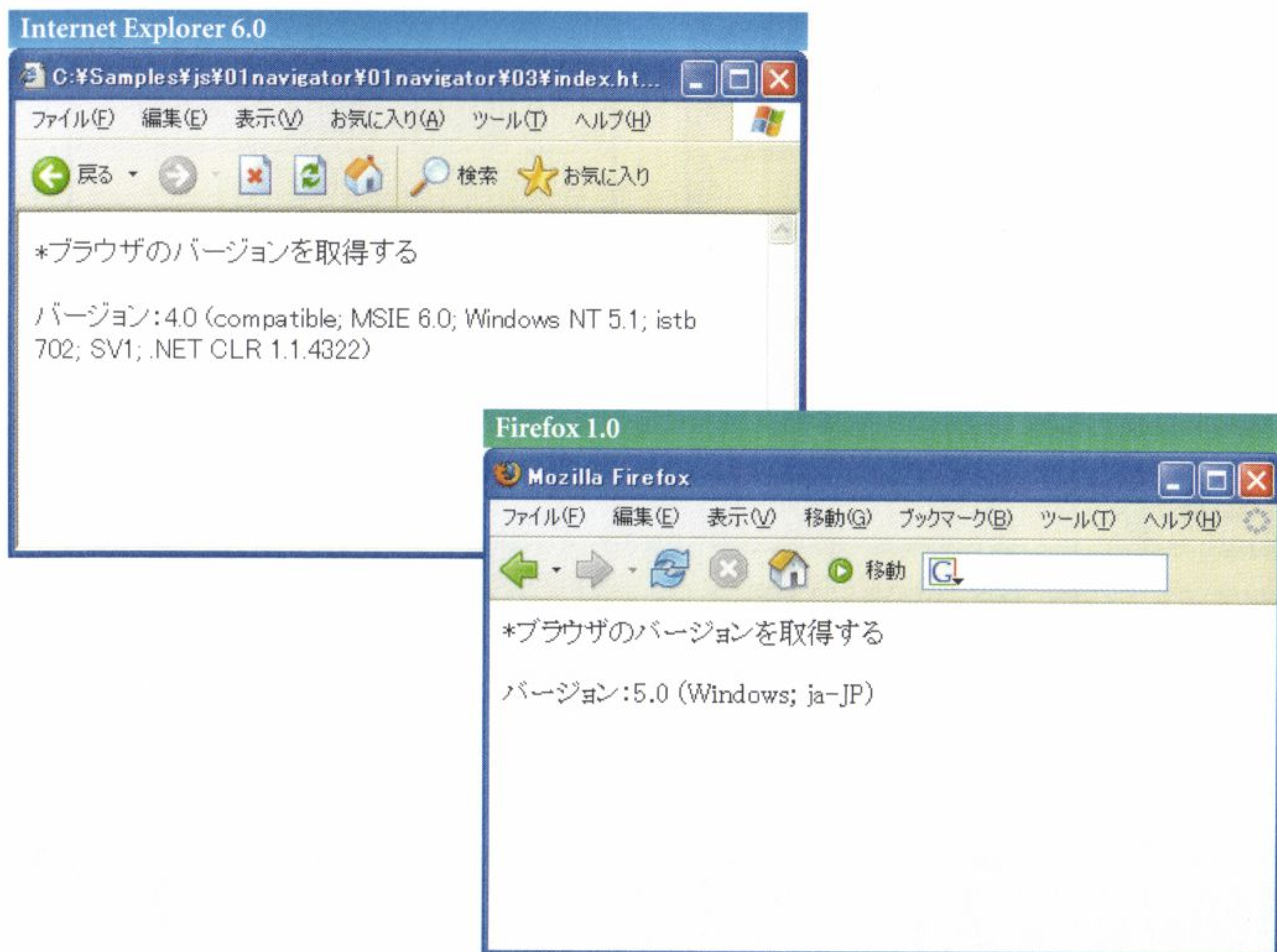
### Sample

```
<script type="text/javascript">
<!--
document.write("コード名:",navigator.appCodeName);
//-->
</script>
```

# ブラウザのバージョンを取得する

## navigator.appVersion [プロパティ]






「appVersion」プロパティは、ブラウザのバージョンの値を持っています。
このバージョン番号は、ブラウザのバージョン番号を正確に表していません。Netscape 6.X や7.X、Mozilla、Firefox、Safariは「5.0」を、Internet Explorer 5.X、6.X や Opera 7.X は「4.0」を返します。また、Internet Explorer 3.X の初期のものは「2.0」を返す場合があります。
バージョン番号の後ろに、OS名、インターナショナル版・U.S版などの種別、CPUの種類などの情報が付加されます。
このプロパティは読み出し専用です。

### Sample

```
<script type="text/javascript">
<!--
document.write("バージョン:",navigator.appVersion);
//-->
</script>
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

JavaScript

# ブラウザのユーザーエージェントを取得する

## navigator.userAgent [プロパティ]

Internet Explorer 6.0




Firefox 1.0




「userAgent」プロパティは、ブラウザのユーザーエージェントの値を持っています。ユーザーエージェントとは、HTTPのヘッダー部分に付けられている文字列です。このプロパティは読み出し専用です。

### Sample

```
<script type="text/javascript">
<!--
document.write("ユーザーエージェント:",navigator.userAgent);
//-->
</script>
```

JavaScript

# プラットホームのタイプを取得する

## navigator.platform [プロパティ]

Internet Explorer 6.0




Firefox 1.0




「platform」プロパティは、プラットフォームのタイプの値を持っています。
サンプルでは、ユーザーの環境に合わせて「Win32」や「MacPPC」といった値を表示します。
JavaScript1.2で追加された、読み出し専用のプロパティです。

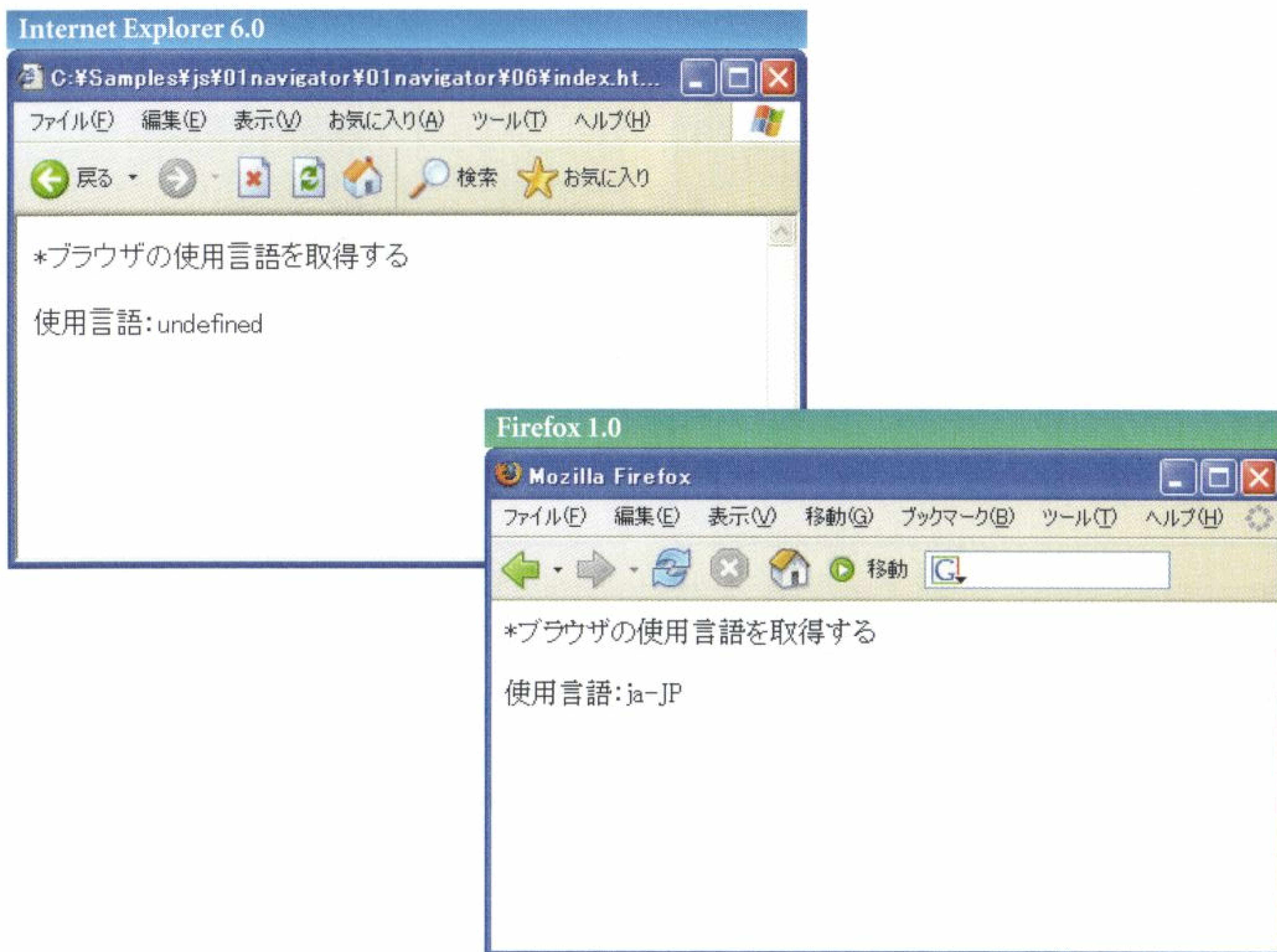
### Sample

```
<script type="text/javascript">
<!--
document.write("お使いのマシンのタイプ:",navigator.platform);
//-->
</script>
```

# ブラウザの使用言語を取得する

## navigator.language [プロパティ]






「language」プロパティは、ブラウザの「ja」や「en」などのLANG属性の値を持っています。[編集]メニューの[設定]－[Navigator]－[言語]から設定することができます。JavaScript1.2で追加された、読み出し専用のプロパティです。
このプロパティは、Internet Explorerでは未対応です。

### Sample

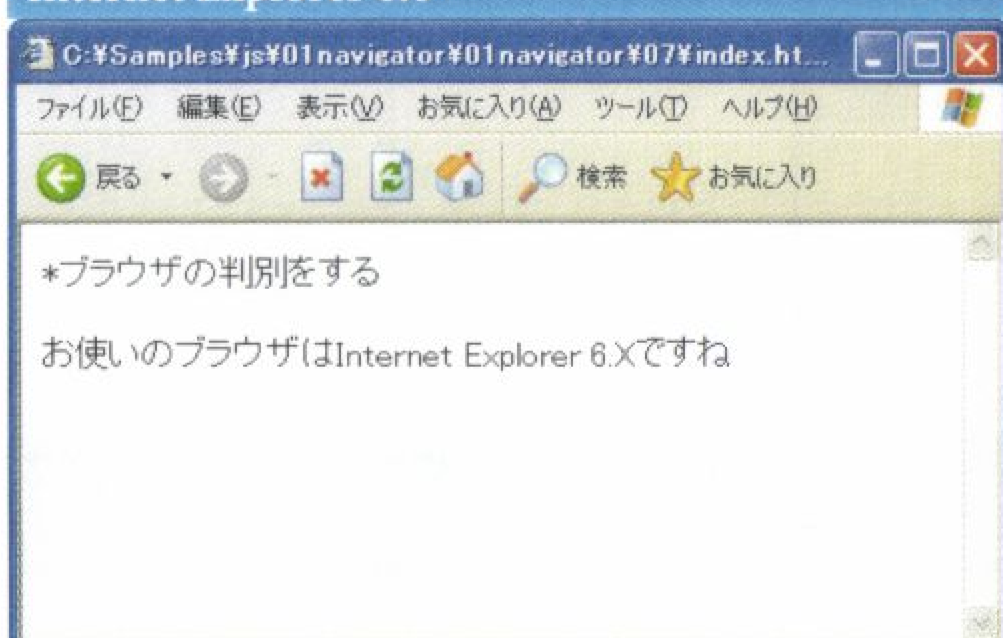
```
<script type="text/javascript">
<!--
document.write("使用言語:",navigator.language);
//-->
</script>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

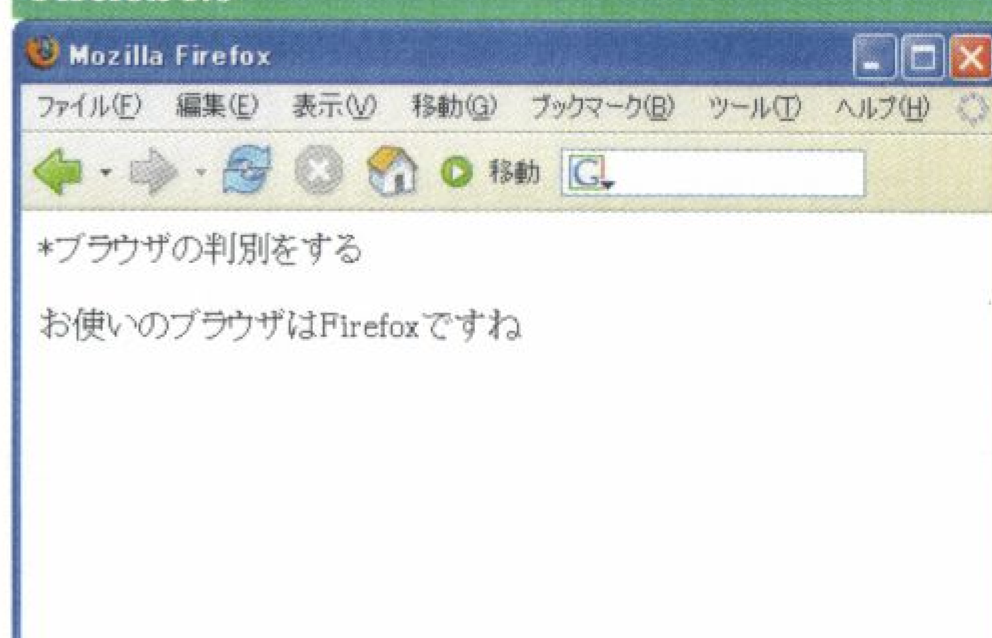
# ブラウザの判別をする

**navigator.appName** [プロパティ]
**navigator.appVersion** [プロパティ]
**navigator.userAgent** [プロパティ]

Internet Explorer 6.0




Firefox 1.0




サンプルでは、まずブラウザ名の1番始めの文字を検索して、それを元にNetscape Navigator、Safari、Firefox、Mozillaと、Internet Explorer、Operaを振り分る処理を行っています。そして次に、バージョンの1番始めの文字やユーザーエージェント内に含まれる文字を検索し、Netscape Navigator、Internet Explorerの各バージョンと、Mozilla、Safari、Firefox、Operaを判断し、各々のブラウザ名を書き出しています。
ブラウザ名の1番始めの文字が「N」のブラウザの中で、Netscape 6.Xや7.X、Mozilla、Safari、Firefoxは、バージョンとして「5.0」を返します。そこでサンプルでは、それらのブラウザのユーザーエージェントに含まれる「Netscape6/」「Netscape/7」「Safari」「Firefox」といったブラウザ固有の文字列を検索して、各々のブラウザを判断しています。
また、ブラウザ名の1番始めの文字が「M」のブラウザのうち、Internet Explorer 3.Xは、バージョンとして「2.0」を返すものがあるので、その点も考慮しています。さらに、Internet Explorer 4.X以外にも、5.Xや6.X、Operaは、現在バージョンとして「4.0」を返します。そこでサンプルでは、バージョン情報の中から、Internet Explorer 5.Xの場合は「MSIE 5」、Internet Explorer 6.Xの場合は「MSIE 6」という文字列を、Operaの場合はユーザーエージェント情報の中から「Opera」という文字列を検索することによって、各々のブラウザを判別しています。

## Sample

```
<script type="text/javascript">
<!--
if( navigator.appName.charAt(0)=="N" ){
    if(navigator.appVersion.charAt(0)==2){ document.write("お使いの
ブラウザはNetscape Navigator 2.Xですね") }
    if(navigator.appVersion.charAt(0)==3){ document.write("お使いの
ブラウザはNetscape Navigator 3.Xですね") }
```

ナビゲータ情報を利用する

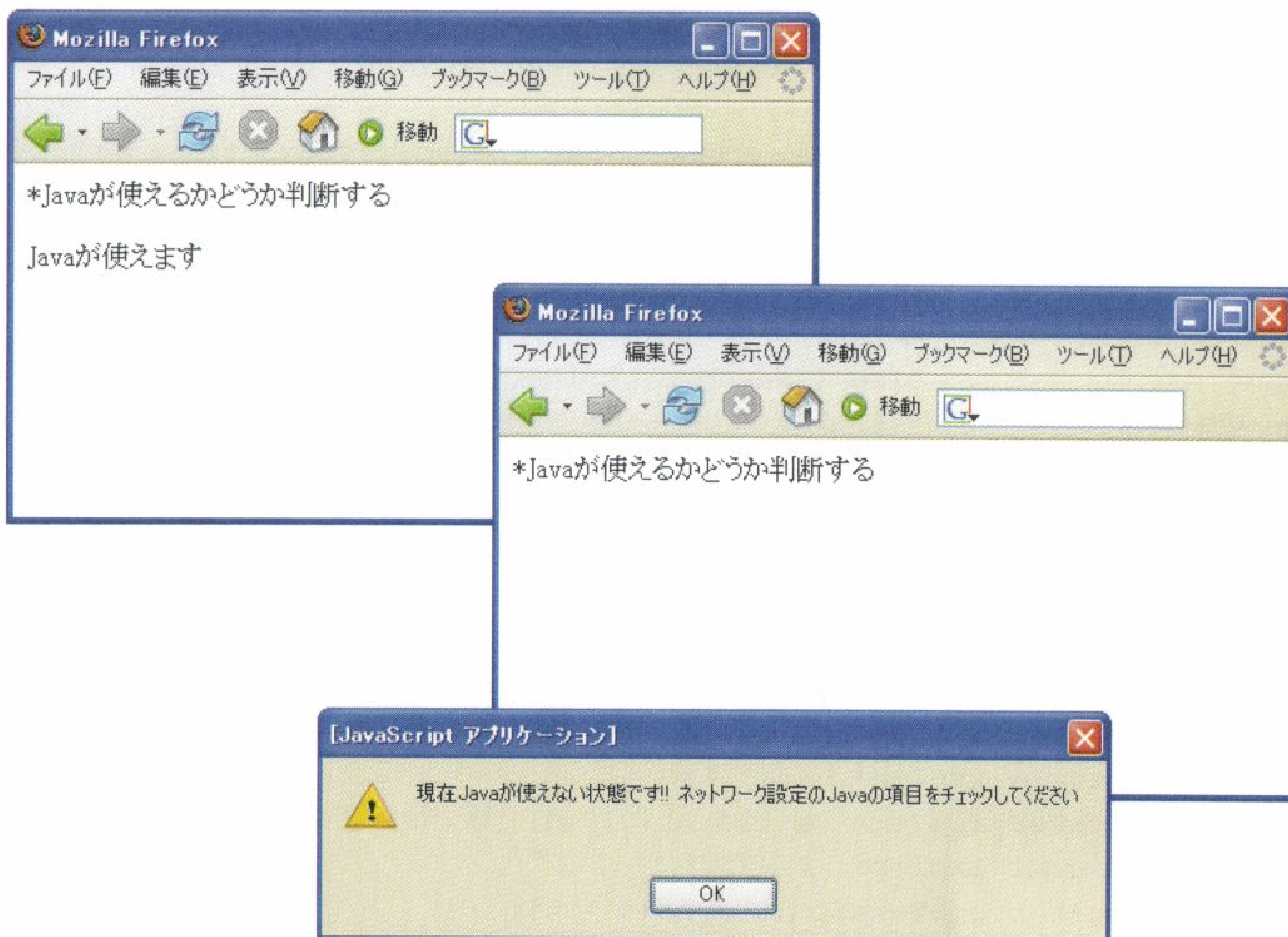
```
    if(navigator.appVersion.charAt(0)==4){ document.write("お使いの
ブラウザはNetscape Navigator 4.Xですね") }
    if(navigator.appVersion.charAt(0)==5){
        if (navigator.userAgent.indexOf("Netscape6/") != -1){docu
ment.write("お使いのブラウザはNetscape Navigator 6.Xですね") }
        else {
            if (navigator.userAgent.indexOf("Netscape/7") != -1){
document.write("お使いのブラウザはNetscape Navigator 7.Xですね") }
            else {
                if (navigator.userAgent.indexOf("Safari") != -1){
document.write("お使いのブラウザはSafariですね") }
                else {
                    if (navigator.userAgent.indexOf("Firefox") !=
-1){document.write("お使いのブラウザはFirefoxですね") }
                    else { document.write("お使いのブラウザはMozilla1.
xですね") }
                }
            }
        }
    }
}
if( navigator.appName.charAt(0)=="M" ){
   if(navigator.appVersion.charAt(0)==2){ document.write("お使いの
ブラウザはInternet Explorer 3.Xですね") }
   if(navigator.appVersion.charAt(0)==3){ document.write("お使いの
ブラウザはInternet Explorer 3.Xですね") }
   if(navigator.appVersion.charAt(0)==4){
        if (navigator.userAgent.indexOf("Opera") != -1){document.
write("お使いのブラウザはOperaですね") }
        else {
            if (navigator.appVersion.indexOf("MSIE 6") != -1){doc
ument.write("お使いのブラウザはInternet Explorer 6.Xですね") }
            else {
                if (navigator.appVersion.indexOf("MSIE 5") !=
-1){document.write("お使いのブラウザはInternet Explorer 5.Xですね") }
                else {document.write("お使いのブラウザはInternet Explo
rer 4.Xですね") }
            }
        }
    }
}
//-->
</script>
```


charAt():「stringオブジェクト」の「n番目の文字を抜き出す」(P.555)
indexOf():「stringオブジェクト」の「先頭から文字列を検索する」(P.558)

# Javaが使えるかどうか判断する

## navigator.javaEnabled() [メソッド]





ブラウザの設定でJavaを使えないようにした場合

サンプルでは、「javaEnabled()」メソッドでJavaが使える状態になっているかを判断し、使えれば「Javaが使えます」と書き出し、そうでなければ警告用のダイアログボックスを開きます。
JavaScript1.1で追加されたメソッドです。

### Sample

```
<script type="text/javascript">
<!--
if (navigator.javaEnabled()) { document.write("Javaが使えます") }
    else window.alert("現在Javaが使えない状態です!! ネットワーク設定のJava
の項目をチェックしてください");
//-->
</script>
```

alert():「windowオブジェクト」の「警告用のダイアログボックスを開く」(P.333)

# 使用可能なMIMEのタイプを取得する

**navigator.mimeTypes** [オブジェクト(配列)]
**navigator.mimeTypes[n].type** [プロパティ]
**navigator.mimeTypes[n].description** [プロパティ]
**navigator.mimeTypes[n].suffixes** [プロパティ]






サンプルでは、「length」プロパティでMIMEタイプの数を出し、そのブラウザで利用可能なMIMEタイプの一覧を作成しています。
「type」プロパティはMIMEのタイプを、「description」プロパティはその詳細を、「suffixes」プロパティは拡張子をそれぞれ返します。
mimeTypesオブジェクトには、その他にも「enablePlugin」(プラグインを使うための名前を返す)というプロパティがあります。
JavaScript1.1で追加された、読み出し専用のプロパティです。

## Sample

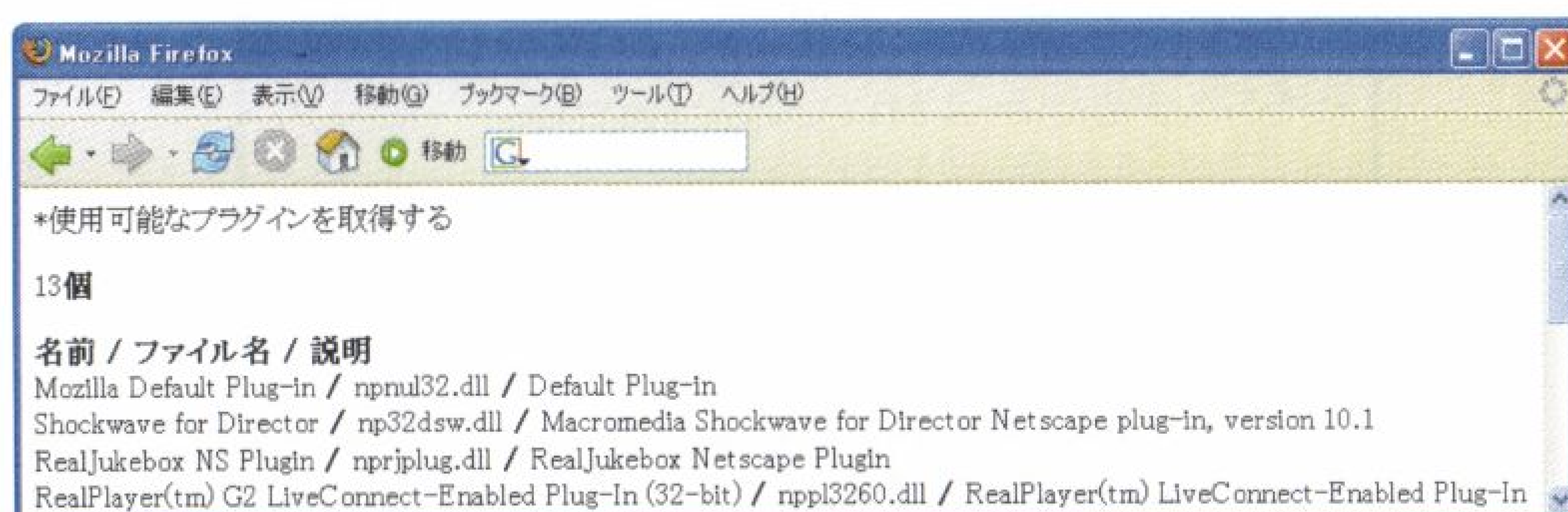
```
<script type="text/javascript">
<!--
var L = navigator.mimeTypes.length;
document.write( L );
document.write("個".bold());
document.write("<p>");
document.write("タイプ / 説明 / 拡張子".bold());
document.write("<br>");
for(i=0; i<L; i++){
document.write(navigator.mimeTypes[i].type);
document.write(" / ".bold());
document.write(navigator.mimeTypes[i].description);
document.write(" / ".bold());
document.write(navigator.mimeTypes[i].suffixes);
document.write("<br>");
}
//-->
</script>
```


bold()：「stringオブジェクト」の「太字（ボールド）にする」（P.547）
length：「複数のオブジェクトで使用できるプロパティ・メソッド」の「オブジェクト（配列）の数を取得する」（P.580）

# 使用可能なプラグインを取得する

**navigator.plugins** [オブジェクト(配列)]
**navigator.plugins[n].name** [プロパティ]
**navigator.plugins[n].filename** [プロパティ]
**navigator.plugins[n].description** [プロパティ]






サンプルでは、「length」プロパティでプラグインの数を出し、そのブラウザで使用可能なプラグインの一覧を作成しています。
「name」プロパティはプラグインの名前を、「filename」プロパティはファイル名を、「description」プロパティはその詳細をそれぞれ返します。
JavaScript1.1で追加された、読み出し専用のプロパティです。



## Sample

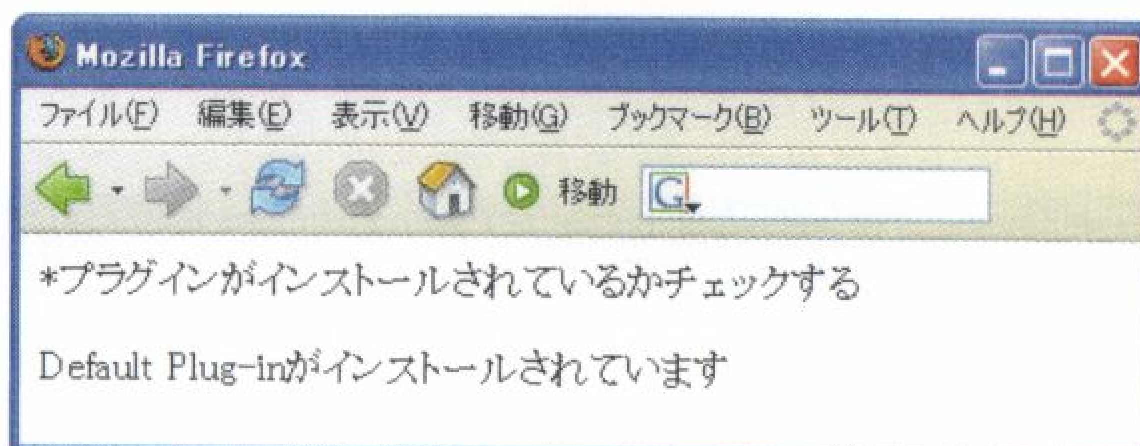
```
<script type="text/javascript">
<!--
var L = navigator.plugins.length;
document.write( L );
document.write("個".bold());
document.write("<p>");
document.write("名前 / ファイル名 / 説明".bold());
document.write("<br>");
for(i=0; i<L; i++){
document.write(navigator.plugins[i].name);
document.write(" / ".bold());
document.write(navigator.plugins[i].filename);
document.write(" / ".bold());
document.write(navigator.plugins[i].description);
document.write("<br>");
}
//-->
</script>
```


bold():「stringオブジェクト」の「太字(ボールド)にする」(P.547)
length:「複数のオブジェクトで使用できるプロパティ・メソッド」の「オブジェクト(配列)の数を取得する」(P.580)

# プラグインがインストールされているかチェックする

**navigator.plugins** [オブジェクト(配列)]
**navigator.plugins[*n*].name** [プロパティ]






サンプルでは、プラグイン配列の中に「Default Plug-in」の一部である「efault」という文字列が含まれているかどうかを検索して、含まれていれば文字を書き出し、そうでなければ警告用のダイアログボックスを開いています。
JavaScript1.1で追加された、読み出し専用のプロパティです。

## Sample



```
<script type="text/javascript">
<!--
function CPlug(CP) {
    for(i=0; i<navigator.plugins.length; i++) {
        if (navigator.plugins[i].name.indexOf(CP) != -1) return
true;
     }
     return false;
}
//-->
</script>
  ～中略～
<script type="text/javascript">
<!--
if (CPlug("efault")) { document.write("Default Plug-inがインストールさ
れています") }
    else window.alert("Default Plug-inがインストールされていません!!");
//-->
</script>
```



alert()：「windowオブジェクト」の「警告用のダイアログボックスを開く」(P.333)
indexOf()：「stringオブジェクト」の「先頭から文字列を検索する」(P.558)
length：「複数のオブジェクトで使用できるプロパティ・メソッド」の「オブジェクト(配列)の数を取得する」(P.580)

# その他のnavigatorオブジェクト

## refresh() [メソッド]

「refresh()」メソッドは、plugins配列をリフレッシュします。この時に真(true)を返すと、それ以降<embed>で指定してプラグインを使用することができ、偽(false)を返した時は、プラグインの配列の作り直しのみを行います。
JavaScript1.1から追加された、pluginsオブジェクトのメソッドです。

**Sample**

```
navigator.plugins.refresh([true|false])
```

## preference() [メソッド]

「preference()」メソッドは、クライアントのpreferenceの設定を行います。
Signed Script内で使用可能です。
JavaScript1.2から追加されたメソッドです。

**Sample**

```
navigator.preference(設定名)
navigator.preference(設定名, 設定値)
```

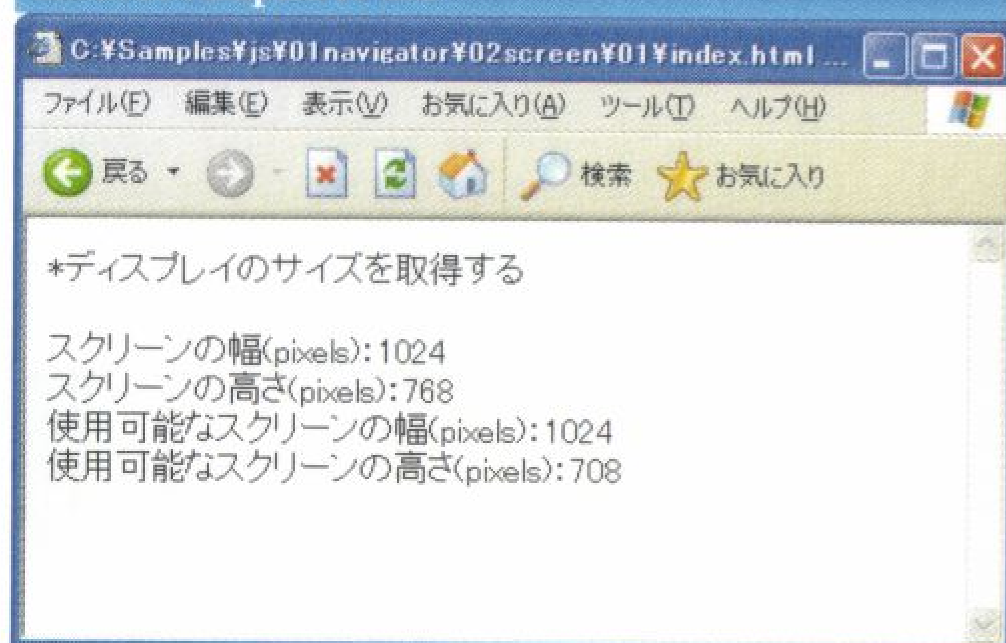
| 設定項目 | 設定名 | 設定値 |
|---|---|---|
| Automatically load images | general.always_load_images | true ¦ false |
| Enable Java | security.enable_java | true ¦ false |
| Enable JavaScript | javascript.enabled | true ¦ false |
| Enable style sheets | browser.enable_style_sheets | true ¦ false |
| Enable autoinstall | autoupdate.enabled | true ¦ false |
| Accept all cookies | network.cookie.cookieBehavior | 0 |
| Accept only cookies that getsent back to the originating server | network.cookie.cookieBehavior | 1 |
| Disable cookies | network.cookie.cookieBehavior | 2 |
| Warn before accepting cookie | network.cookie.warnAboutCookies | true ¦ false |

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

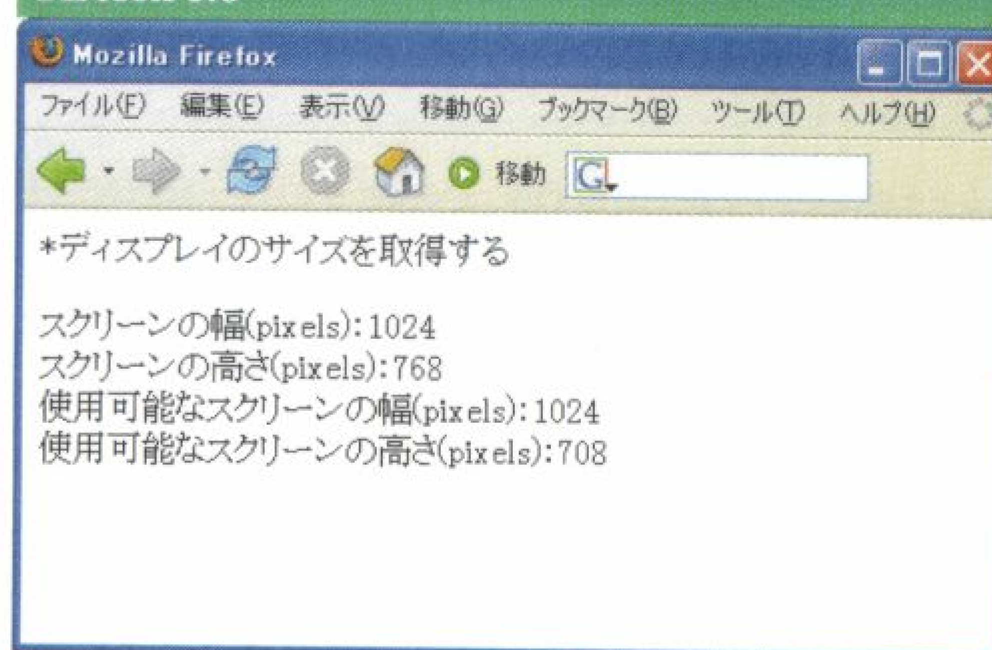
# ディスプレイのサイズを取得する

screen.width [プロパティ]
screen.height [プロパティ]
screen.availWidth [プロパティ]
screen.availHeight [プロパティ]

Internet Explorer 6.0




Firefox 1.0




「width」プロパティと「height」プロパティは、スクリーン全体の幅と高さの値を返します。「availWidth」プロパティと「availHeight」プロパティは、スクリーン全体からWindowsのタスクバーのように表示できない部分を除いたスクリーンの幅と高さの値を、それぞれ返します。

Macintosh版のInternet Explorerでは、「screen.availWidth」「screen.availHeight」が正常な値を返さない場合があります。

JavaScript1.2で追加された、読み出し専用のプロパティです。

## Sample

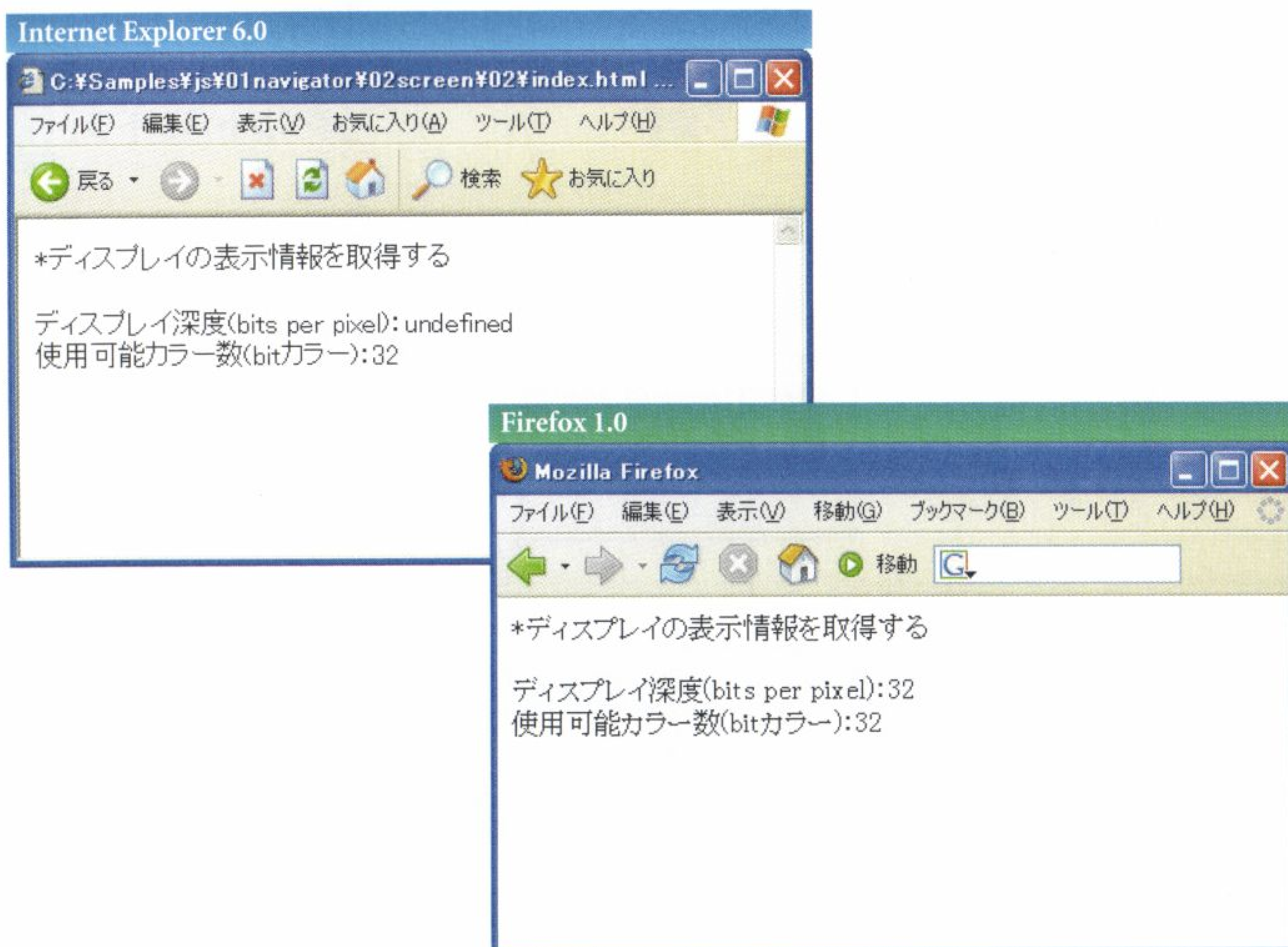
```
<script type="text/javascript">
<!--
document.write("スクリーンの幅(pixels):",screen.width);
document.write("<br>");
document.write("スクリーンの高さ(pixels):",screen.height);
document.write("<br>");
document.write("使用可能なスクリーンの幅(pixels):",screen.availWidth);
document.write("<br>");
document.write("使用可能なスクリーンの高さ(pixels):",screen.availHeight);
//-->
</script>
```

ディスプレイ情報を利用する

# ディスプレイの表示情報を取得する

**screen.pixelDepth** [プロパティ]
**screen.colorDepth** [プロパティ]






「pixelDepth」プロパティは1ピクセルあたり何ビットの情報で表示するかの値を、「colorDepth」プロパティは表示可能色数の値を、それぞれビット数で返します。
たとえば、256色の場合は8、65,000色(Macintoshでは32,000色)の場合は16となります。
JavaScript1.2で追加された、読み出し専用のプロパティです。
Internet Explorerでは、「pixelDepth」プロパティに未対応です。

## Sample

```
<script type="text/javascript">
<!--
document.write("ディスプレイ深度(bits per pixel):",screen.pixelDepth);
document.write("<br>");
document.write("使用可能¥カラー数(bitカラー):",screen.colorDepth);
//-->
</script>
```

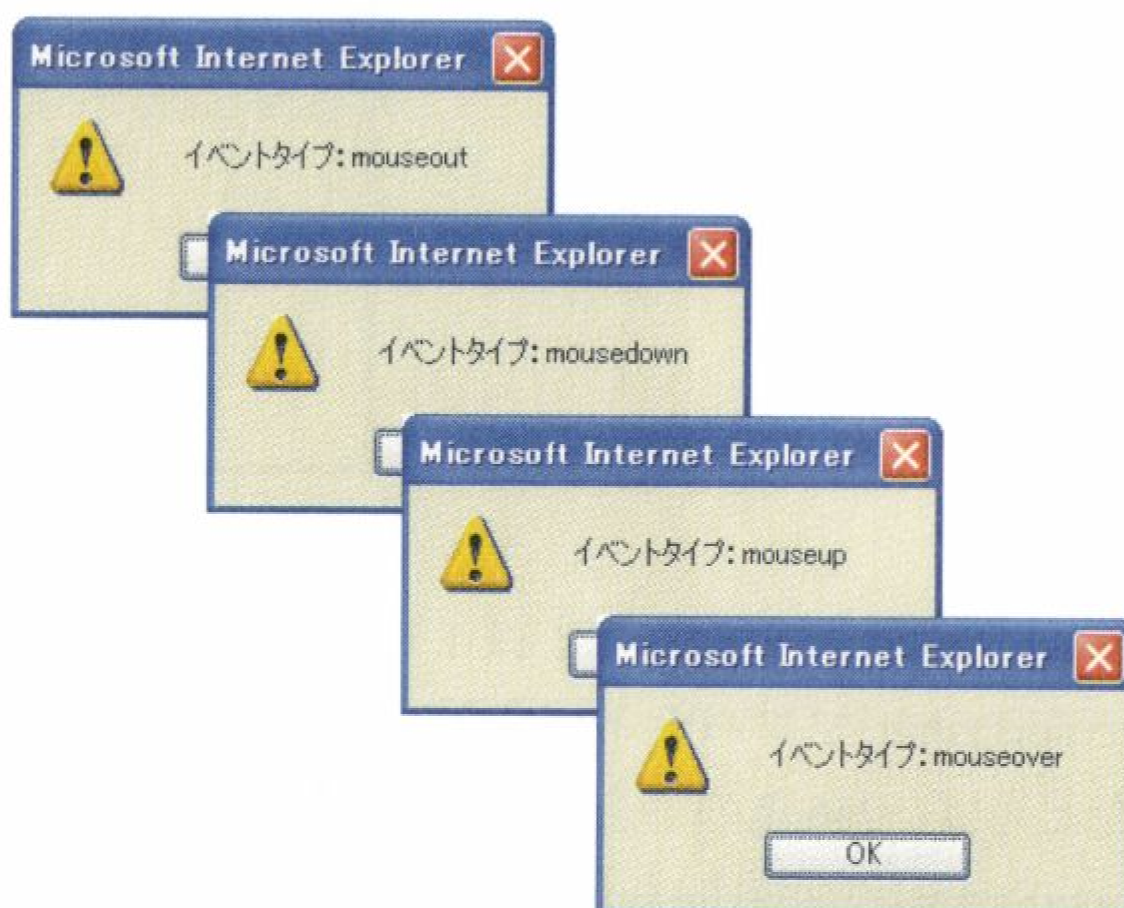
# イベントのタイプを取得する

**event.type** イベントの種類　[プロパティ]
**onClick="スクリプト | 関数"**　[イベントハンドラ]
**onMouseOut="スクリプト | 関数"**　[イベントハンドラ]
**onMouseDown="スクリプト | 関数"**　[イベントハンドラ]
**onMouseUp="スクリプト | 関数"**　[イベントハンドラ]
**onMouseOver="スクリプト | 関数"**　[イベントハンドラ]










「type」プロパティは、発生したイベントのタイプの値を持っています。
サンプルでは、フォームやリンクに色々なイベントハンドラを設定し、イベントが発生した時に警告用のダイアログボックスにそのイベントタイプを表示しています。
「onClick」は、オブジェクトがクリックされた時にイベントを発生し、イベントタイプは「click」を持ちます。同様に、「onMouseOut」はオブジェクトからマウスカーソルが離れた時に発生し「mouseout」を、「onMouseDown」はマウスボタンが押し下げられた時に発生し「mousedown」を、「onMouseUp」はマウスボタンが離された時に発生し「onmouseup」を、「onMouseOver」はオブジェクト上にマウスポインタが来た時に発生し「onmouseover」を、それぞれ持っています。
JavaScript1.2で追加されたプロパティです。

**Sample**

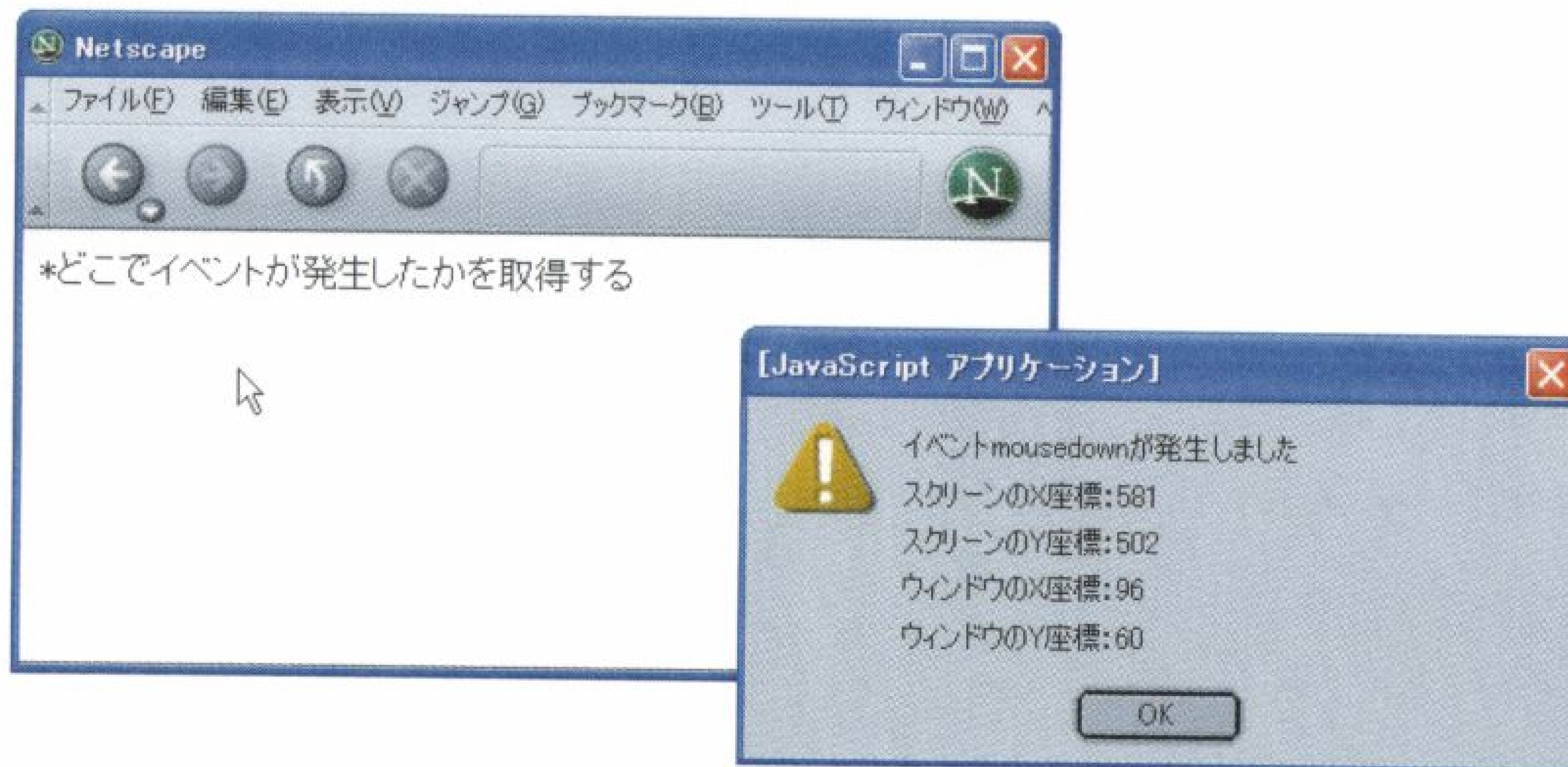
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*イベントのタイプを取得する
<p>
<form>
 <input type="button" value=" onClick Event!! " onClick="alert('イベントタイプ:'+ event.type)">
</form>
</p>
<p>
<a href="#" onMouseOut="alert('イベントタイプ:'+ event.type)">
onMouseOut Event!!
</a>
</p>
<p>
<a href="#" onMouseDown="alert('イベントタイプ:'+ event.type)">
onMouseDown Event!!
</a>
</p>
<p>
<a href="#" onMouseUp="alert('イベントタイプ:'+ event.type)">
onMouseUp Event!!
</a>
</p>
<p>
<a href="#" onMouseOver="alert('イベントタイプ:'+ event.type)">
onMouseOver Event!!
</a>
</p>
</body>
</html>
```


イベントタイプ一覧:リファレンス「イベントタイプ」(P.568)

# どこでイベントが発生したかを取得する

| | | |
|---|---|---|
| イベント.type | イベントの種類 | [プロパティ] |
| イベント.screenX | ディスプレイのX軸の値 | [プロパティ] |
| イベント.screenY | ディスプレイのY軸の値 | [プロパティ] |
| イベント.pageX | ウィンドウのX軸の値 | [プロパティ] |
| イベント.pageY | ウィンドウのY軸の値 | [プロパティ] |
| オブジェクト.onmousedown=スクリプト / 関数 | | [イベント] |






「screenX」プロパティと「screenY」プロパティはイベントが起こったディスプレイ上のX軸とY軸の値を、「pageX」プロパティと「pageY」プロパティはイベントが起こったウィンドウの表示領域上のX軸とY軸の値を、それぞれ持っています。

サンプルでは、マウスボタンが押された時に、イベントタイプ「document.onmousedown」によってイベントを取得して関数「eve1()」を発生させ、関数の処理で発生したイベントが持っているプロパティの値を取り出し、それを警告用のダイアログボックスに表示しています。

JavaScript1.2で追加された用法です。

Internet Explorerは、イベントの取り扱い方法にNetscape Navigatorと違いがあります。「Internet Explorerでのeventオブジェクトの使用法」(右ページ)も合わせて参照してください。

## Sample

```
<script type="text/javascript">
<!--
function eve1(e) {
    alert (     "イベント"+e.type +"が発生しました" + "¥n" +
                "スクリーンのX座標:" + e.screenX + "¥n" +
                "スクリーンのY座標:" + e.screenY + "¥n" +
                "ウィンドウのX座標:" + e.pageX + "¥n" +
                "ウィンドウのY座標:" + e.pageY );
  return true;
```

```
}
document.onmousedown = eve1;
//-->
</script>
```

イベントタイプ一覧：リファレンス「イベントタイプ」(P.622)

TIPS

## Internet Explorerでのeventオブジェクトの使用法

Internet ExplorerのサポートしているJScriptを使用しても、Internet Explorer 4.0からは、Netscapeと同様にeventオブジェクトを使用して、ウィンドウ上のどこでイベントが発生しても、そのイベントを取得できます。しかし、オブジェクトの取り扱い方などに、Netscapeと微妙な違いがあります。

取得するイベントを設定する方法はNetscapeと同じです。しかし、Netscapeのeventオブジェクトが直接イベントからイベントに関する情報を取り出すのに対し、Internet Explorerのeventオブジェクトはwindowオブジェクトのプロパティなので、「window.event.プロパティ」の用法でイベントの情報を取得することができます。

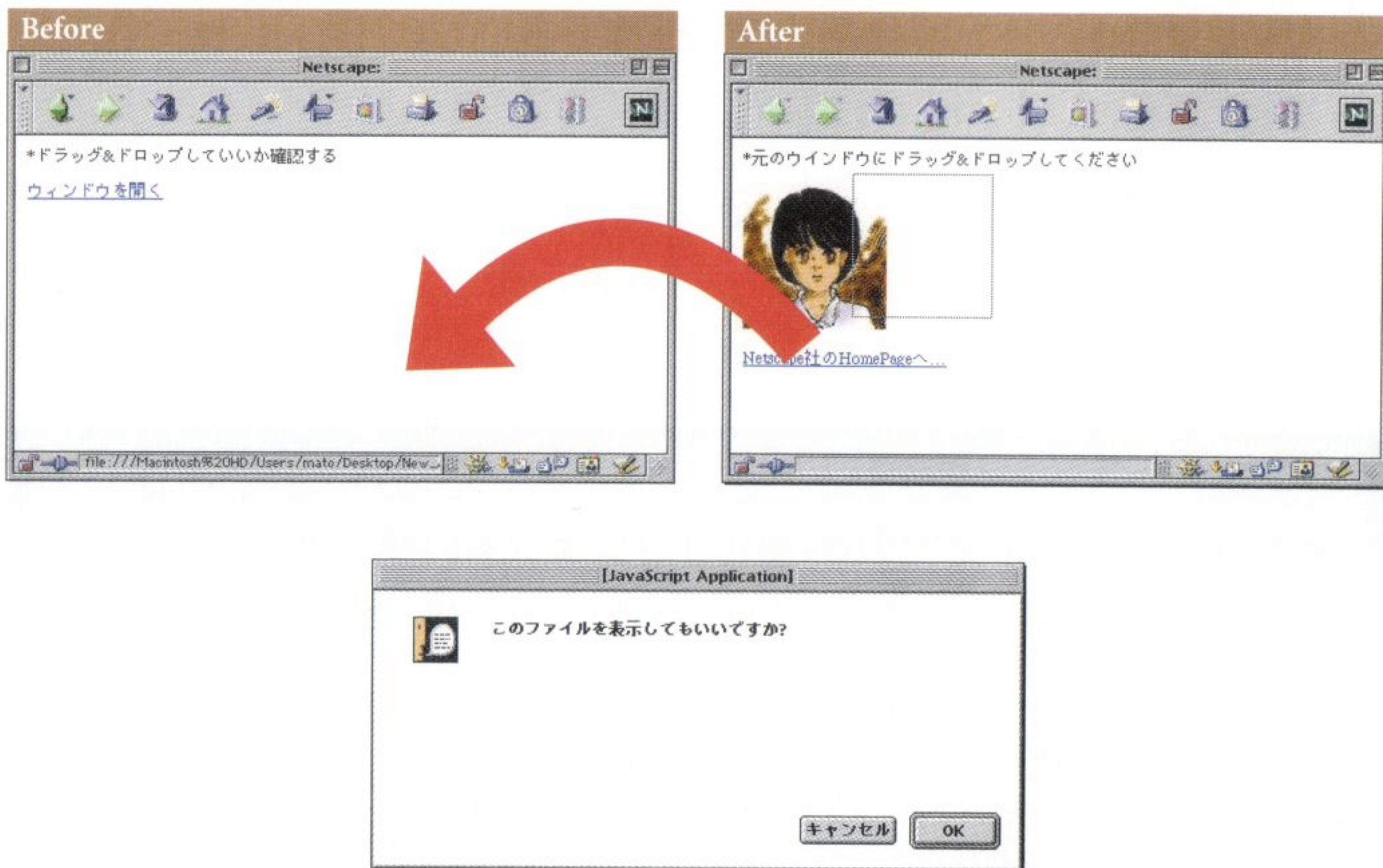
次のサンプルは、「どこでイベントが発生したかを取得する」をInternet Explorerで使用できるようにしたものです（このサンプルの内容は、CD-ROMの中の「js」－「column」－「ie_event」の中に収録してあります）。eventオブジェクトの設定方法の違いに注目してください。また、プロパティにも「pageX」や「pageY」のようにNetscapeのみでサポートされているものや、「offsetX」や「offsetY」のようにInternet Explorerのみでサポートされているものがあるので、注意してください。

```
<script type="text/javascript">
<!--
function eve1() {
     alert ( "イベント"+window.event.type +"が発生しました" + "¥n" +
             "スクリーンのX座標 :" + window.event.screenX + "¥n" +
             "スクリーンのY座標 :" + window.event.screenY + "¥n" +
             "ウィンドウのX座標 :" + window.event.offsetX + "¥n" +
             "ウィンドウのY座標 :" + window.event.offsetY );
    return true;
}
document.onmousedown = eve1;
//-->
</script>
```

# ドラッグ&ドロップしていいか確認する

**オブジェクト.ondragdrop=スクリプト | 関数** [イベント]






イベント「ondragdrop」は、ウィンドウ上で画像やHTMLファイルなどのドラック&ドロップの動作が行われた時のイベントを取得します。
サンプルでは、画像やリンクをウィンドウ上にドロップした時に、「window.ondragdrop」によってイベントを取得して関数「eve1()」を発生させ、関数の処理でファイルを表示していいかどうかを確認するダイアログボックスを開いています。
ただし、Windows 95版のNetscape Navigator 4.Xは、画像のドラッグ&ドロップに対応していません。
JavaScript1.2で追加された用法です。

## Sample

【ドラッグ&ドロップされるウィンドウ】

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function eve1() {
    if ( confirm ( "このファイルを表¥示してもいいですか?") ) {
        return true;
        }
    return false;
}
window.ondragdrop = eve1;
//-->
</script>

</head>
<body>
*ドラッグ&ドロップしていいか確認する
<p>
<a href="NewWin.html" target="_Open">ウィンドウを開く</a>
</p>
</body>
</html>
```

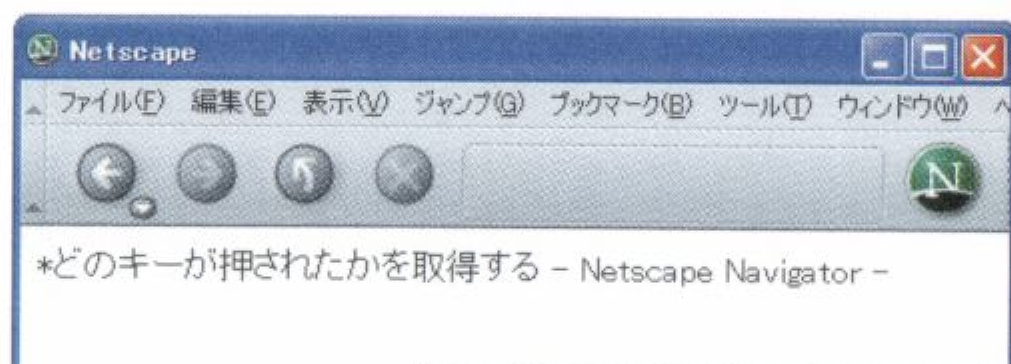
【ドラッグ&ドロップ元のウィンドウ】

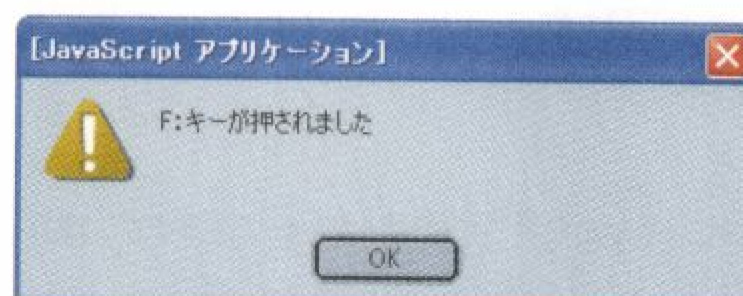
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
</head>
<body>
<p>*元のウインドウにドラッグ&ドロップしてください</p>
<p>
<img src="image.jpg" name="eve04" alt="image.jpg" width="100" hei
ght="100">
</p>
<a href="http://www.netscape.com/">Netscape社のHomePageへ...</a>
</body>
</html>
```

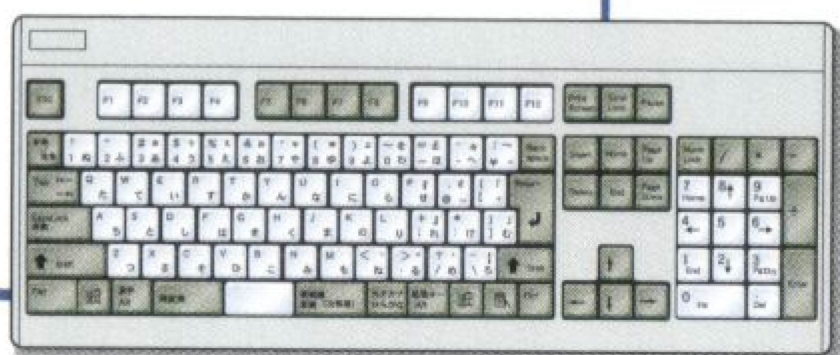
# どのキーが押されたかを取得する

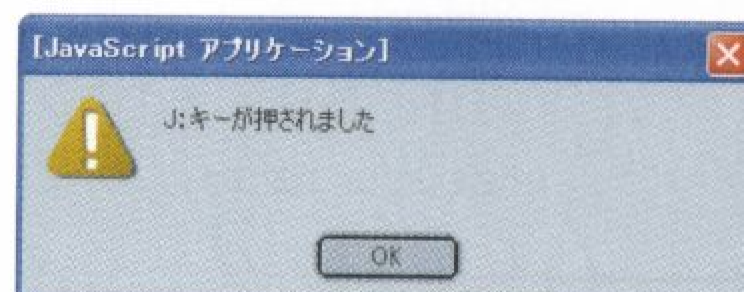
**イベント.which** [プロパティ]
**オブジェクト.onkeydown= スクリプト | 関数** [イベント]








「which」プロパティは、イベント発生時に押されたキーのASCIIの値を持っています。サンプルでは、キーが押されたことを「document.onkeydown」によってイベントとして捉え、「which」プロパティの値を警告用のダイアログボックスに表示しています。
その時、ASCIIの値のままでは「a」のキーは「65」、「b」のキーは「66」といったコード番号なので、それをstringオブジェクトの「fromCharCode」メソッドを使用して英数字に変換しています。
なお、Internet Explorerでキーのコード番号を取得する場合は、「keyCode」プロパティを使い、「window.event.keyCode」とします。
JavaScript1.2で追加された用法です。



## Sample

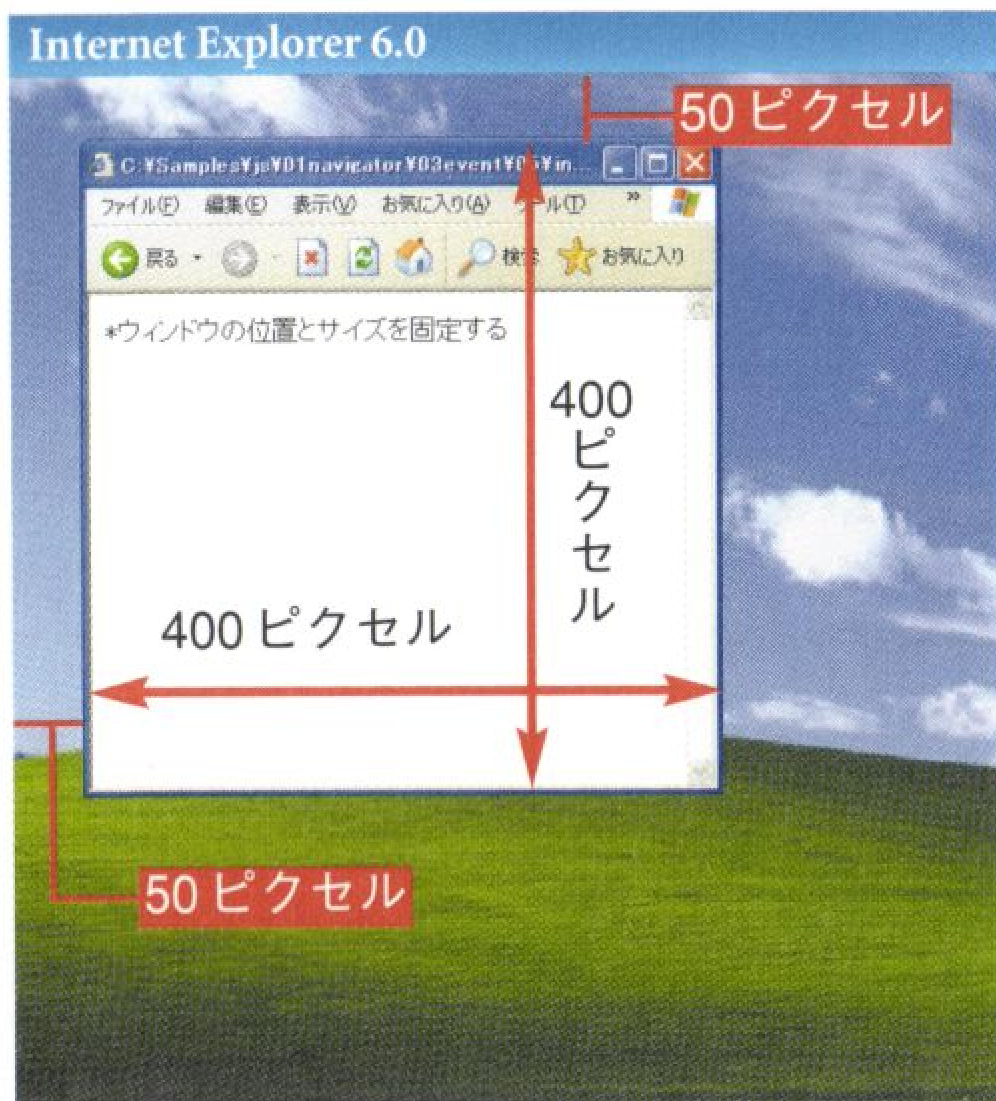
```
<script type="text/javascript">
<!--
function eve1(e) {
    alert ( String.fromCharCode(e.which) + ":キーが押されました");
    return true;
}
document.onkeydown = eve1;
//-->
</script>
```

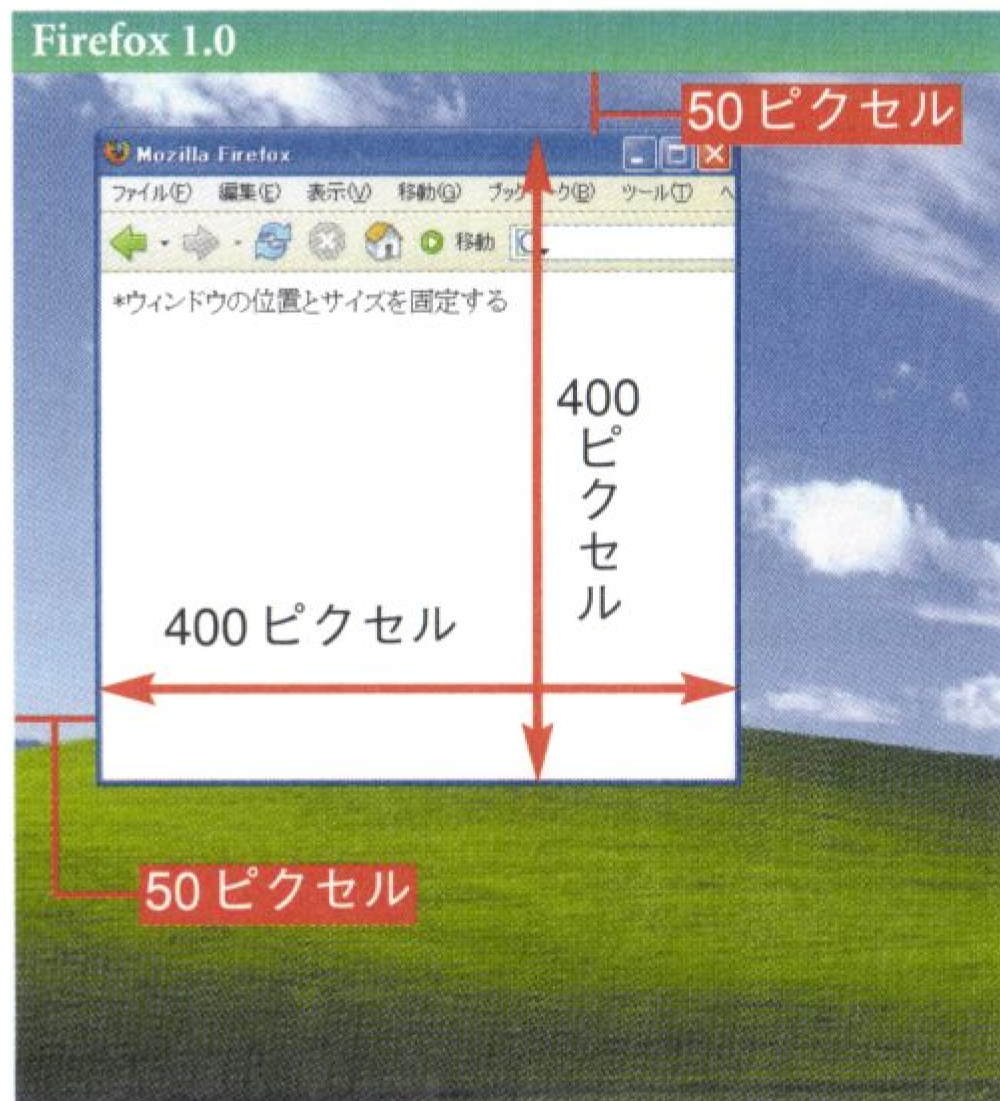

String.fromCharCode():「stringオブジェクト」の「ISO-Latin-1のコード番号を文字に変換する」(P.561)
イベントタイプ一覧:リファレンス「イベントタイプ」(P.622)

# ウィンドウの位置とサイズを固定する

**オブジェクト.onmove= スクリプト | 関数** [イベント]
**オブジェクト.onresize= スクリプト | 関数** [イベント]






イベントの「onmove」はウィンドウが移動した時の、「onresize」はウィンドウのサイズが変更された時のイベントを取得します。
サンプルでは、まずHTMLファイルが読み込まれた時に、ディスプレイの上から50ピクセルで左端から50ピクセルの位置へ、Internet Explorerでは外周が、Netscape Navigatorでは表示領域が、400ピクセル×400ピクセルになるように、ウィンドウの位置とサイズを変更しています。そしてそれ以降、ウィンドウが動かされたり、ウィンドウサイズが変更された時を、それぞれイベントとして取得し、再び同じ位置とサイズにウィンドウを戻しています。
Windows版のNetscape Navigatorでは、スクロールバーが表示された時にも「onresize」イベントが発生するので、注意してください。また、Netscape 6.X以降やInternet Explorerではイベント「onmove」を取得できないので、ウィンドウの位置を固定することはできません。
JavaScript1.2で追加された用法です。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
window.moveTo(50,50);
window.resizeTo(400,400);
function eve1() { window.moveTo(50,50) }
function eve2() { window.resizeTo(400,400) }
window.onmove = eve1;
window.onresize = eve2;
//-->
</script>

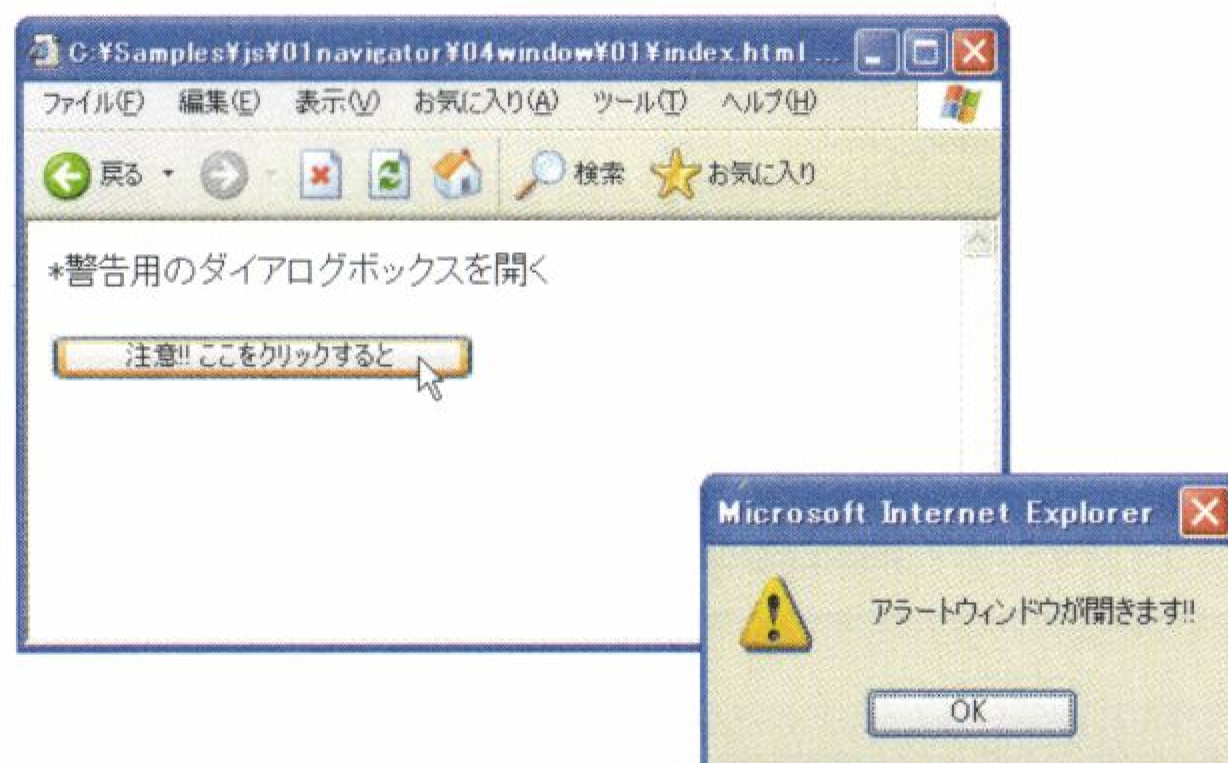
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*ウィンドウの位置とサイズを固定する
</body>
</html>
```


moveTo():「windowオブジェクト」の「ブラウザを指定した位置へ移動する」(P.351)
resizeTo():「windowオブジェクト」の「ブラウザの大きさを指定してリサイズする」(P.354)
イベントタイプ一覧:リファレンス「イベントタイプ」(P.622)

# 警告用のダイアログボックスを開く

alert(文字列) [メソッド]
onClick="スクリプト | 関数" [イベントハンドラ]




「alert()」メソッドは、警告用のダイアログボックスを開きます。
サンプルでは、ボタンをクリックした時にイベントハンドラ「onClick」が関数「EVENT2()」を発生し、ダイアログボックスを開いています。
利用者に注意をうながす時などに使用します。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
    function EVENT1(){ window.alert("アラートウィンドウが開きます!!") }
//-->
</script>
 ～中略～
</head><body>
*警告用のダイアログボックスを開く
<p>
<form>
<input type="button" value=" 注意!! ここをクリックすると " onClick="EVE
NT1()">
</form>
</p>
</body></html>
```

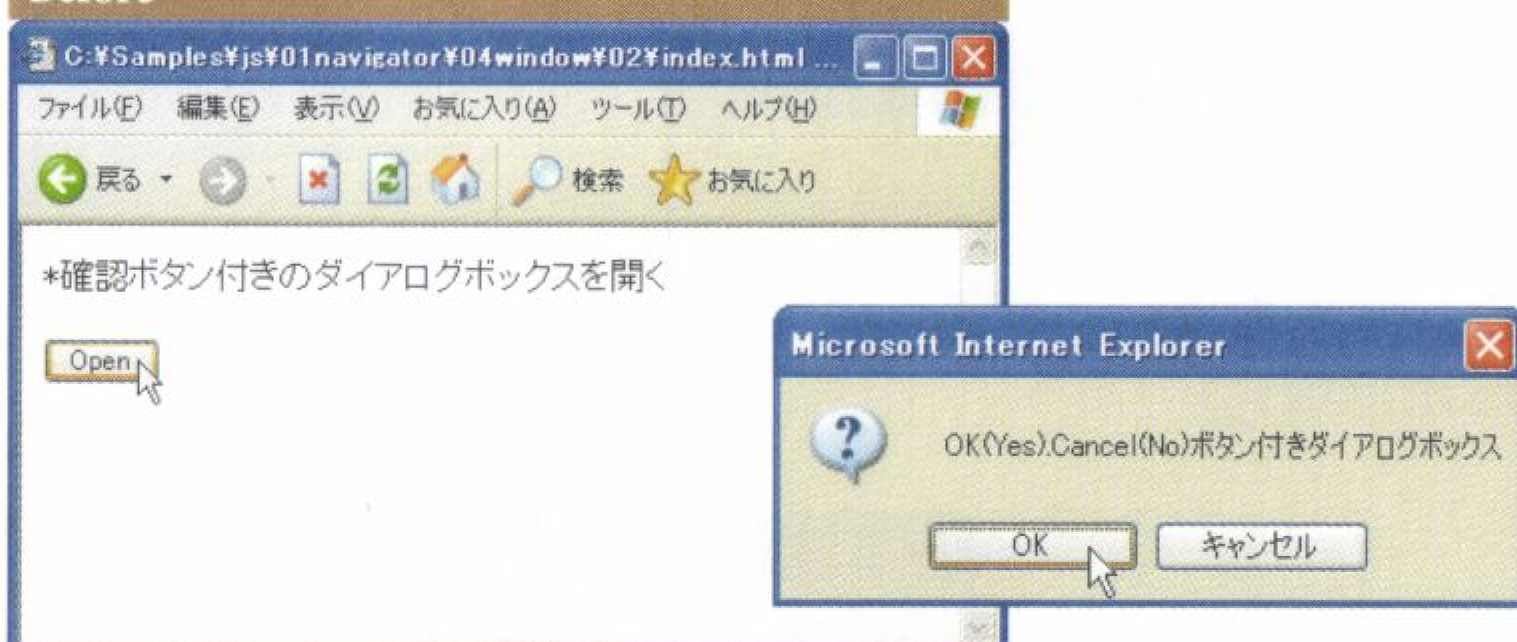

TIPS：「「alert()」メソッドを使いすぎていませんか？」(P.335)

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

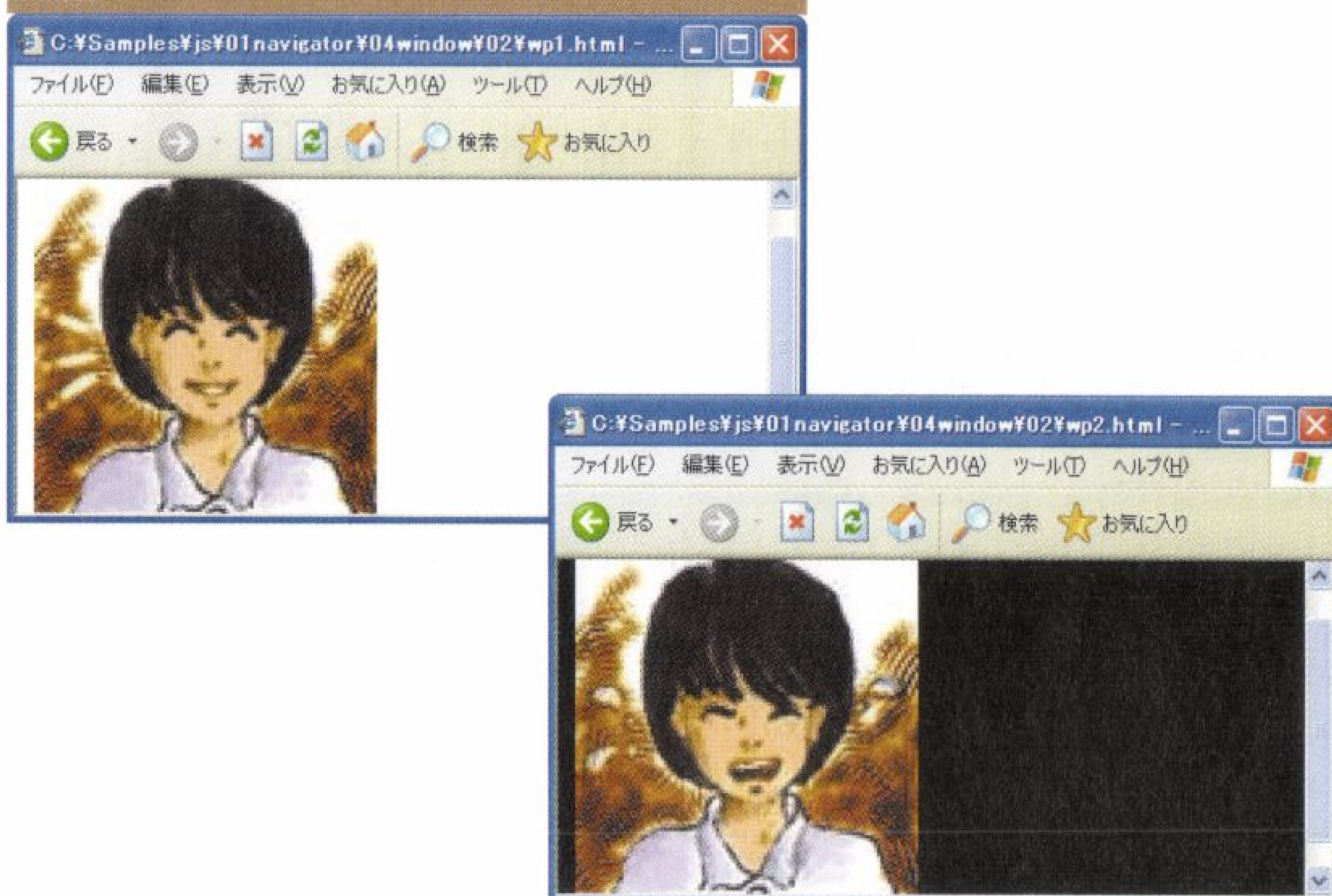
# 確認ボタン付きのダイアログボックスを開く

## confirm(文字列) [メソッド]

Before




After






「confirm()」メソッドは、確認ボタン付きのダイアログボックスを開きます。
確認ボタンの名称は、Windowsでは[OK]と[キャンセル]、Macintoshでは[Yes]と[No]でOSによって違います。
[OK]または[Yes]のボタンが押された時は真(true)の値を、[キャンセル]または[No]のボタンが押された時は偽(false)の値を返します。
サンプルでは、押されたボタンによって違うページがロードされるようになっています。実際に試す場合には、この他にも「wp1.html」と「wp2.html」を用意してください。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function EVENT3(){
    if ( confirm ("OK(Yes).Cancel(No)ボタン付きダイアログボックス") ) {
location.href="wp1.html" }
    else { location.href="wp2.html" }
}
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*確認ボタン付きのダイアログボックスを開く
<p>
<form>
    <input type="button" value=" Open " onClick="EVENT3()">
</form>
</p>
</body>
</html>
```


location.href：「locationオブジェクト」の「自ページのURLを取得する」(P.408)

## TIPS

### 「alert()」メソッドを使いすぎていませんか?

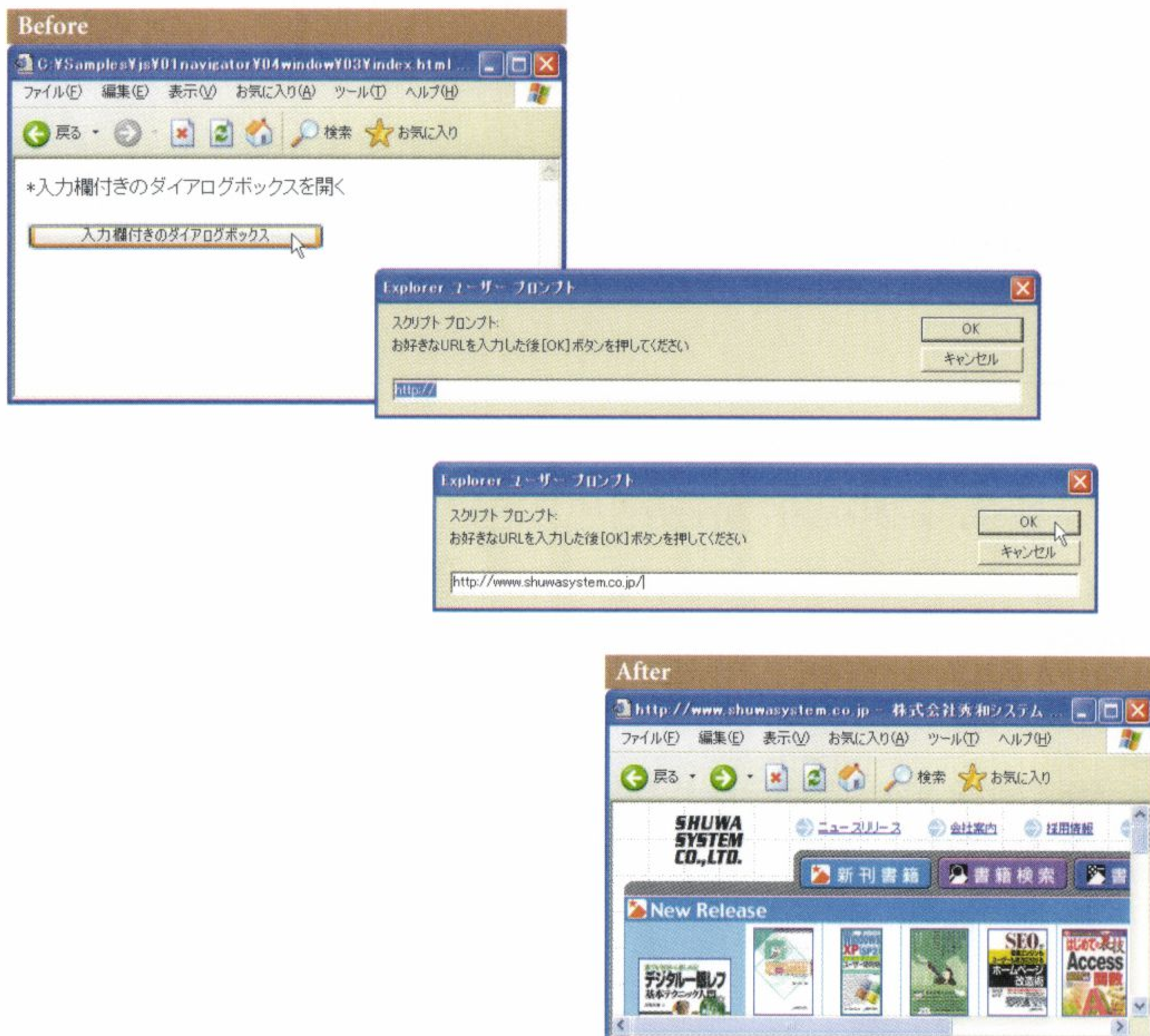
「document.write()」などを使って長い文章を書き出した時に、ブラウザのバージョンによっては、エラーが発生する時があります。

これは、Netscape Navigatorが1行中に1バイト文字で255字まで、2バイト文字だとその半分の文字しか扱えなかったことからくる問題です。

この問題を回避するために、「document.write("文字列A"+"文字列B")」といった具合に、JavaScriptで書き出す文章は短く分けることをお勧めします。

# 入力欄付きのダイアログボックスを開く

**prompt(*文字列*,*値*)** [メソッド]




「prompt()」メソッドは、入力欄付きのダイアログボックスを開きます。
「prompt(ウィンドウ内に表示する文字列,入力ボックス内の初期値)」と指定します。[OK]ボタンが押されると入力欄付きダイアログボックス内の値が代入され、[キャンセル]ボタンが押されるとnullの値を返します。
サンプルでは、入力欄に何も入力されていなかったり、初期値のままだったり、[キャンセル]ボタンが押された時には、「alert()」メソッドで警告用のダイアログボックスを開きます。それ以外の場合は入力されたURLをロードし、もしその時に入力されたURLが不正な場合はブラウザ自身が警告をします。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function EVENT4(){
    PRO=prompt("お好きなURLを入力した後[OK]ボタンを押してください","http://");
        if (!(PRO=="" || PRO==null || PRO=="http://")) { location.href=PRO }
        else { alert("なにも入力されていないか、[Cancel]ボタンが押されました") }
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*入力欄付きのダイアログボックスを開く
<p>
<form>
    <input type="button" value=" 入力欄付きのダイアログボックス " onClick="EVENT4()">
</form>
</p>
</body>
</html>
```

location.href：「locationオブジェクト」の「自ページのURLを取得する」(P.408)

# 新しいウィンドウを開く

**window.open("*URL*","*ウィンドウ名*","*属性*")** ［メソッド］

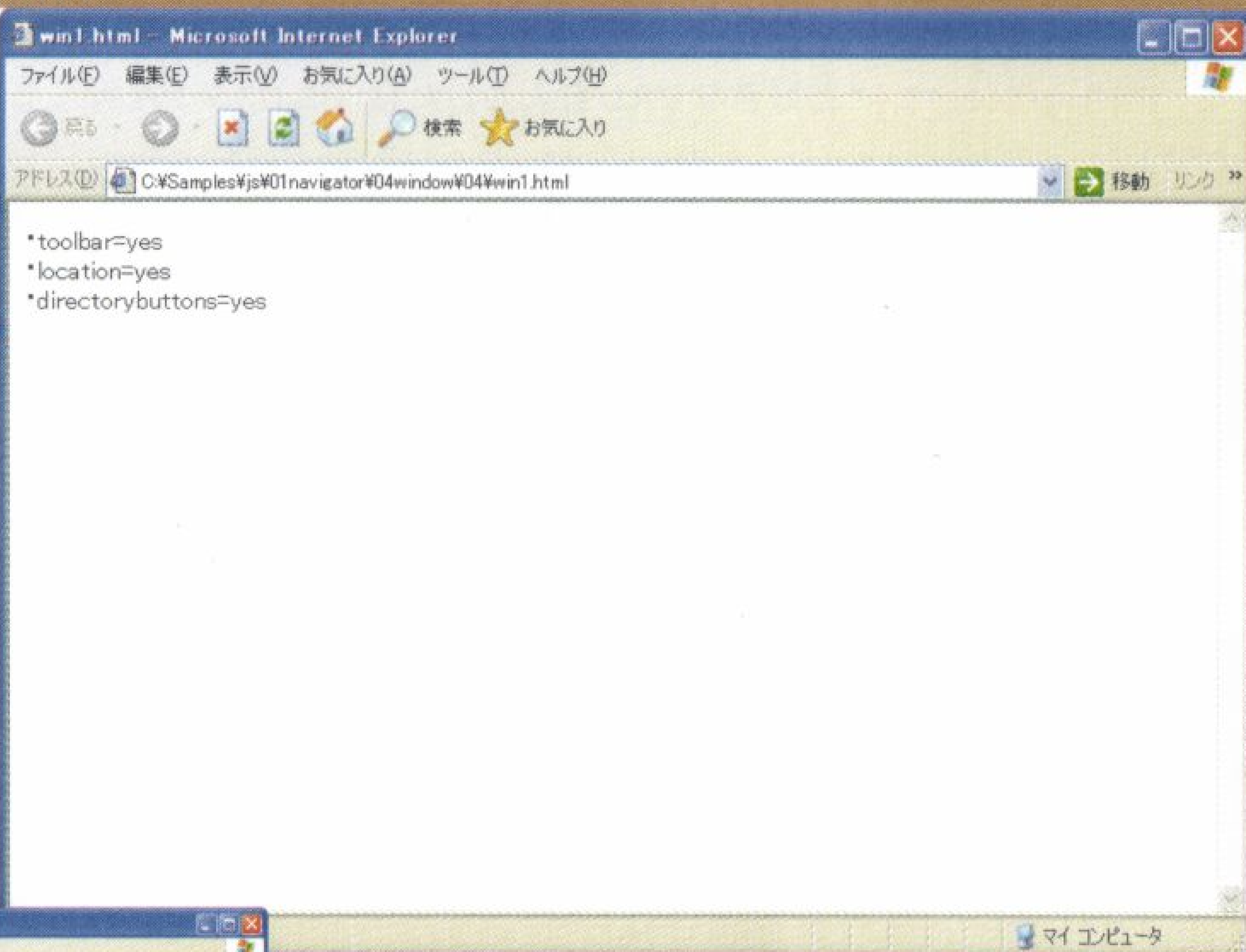
Before

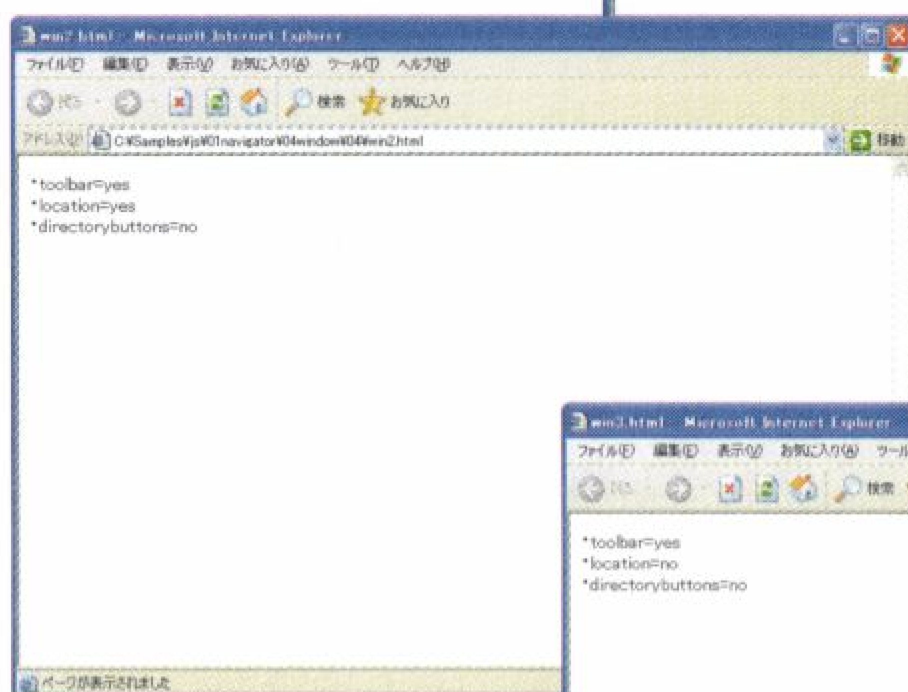



*新しいウィンドウを開く

全部あり
ディレクトリなし
ディレクトリ・ロケーションなし
ディレクトリ・ロケーション・ツールバーなし
サイズ指定有り

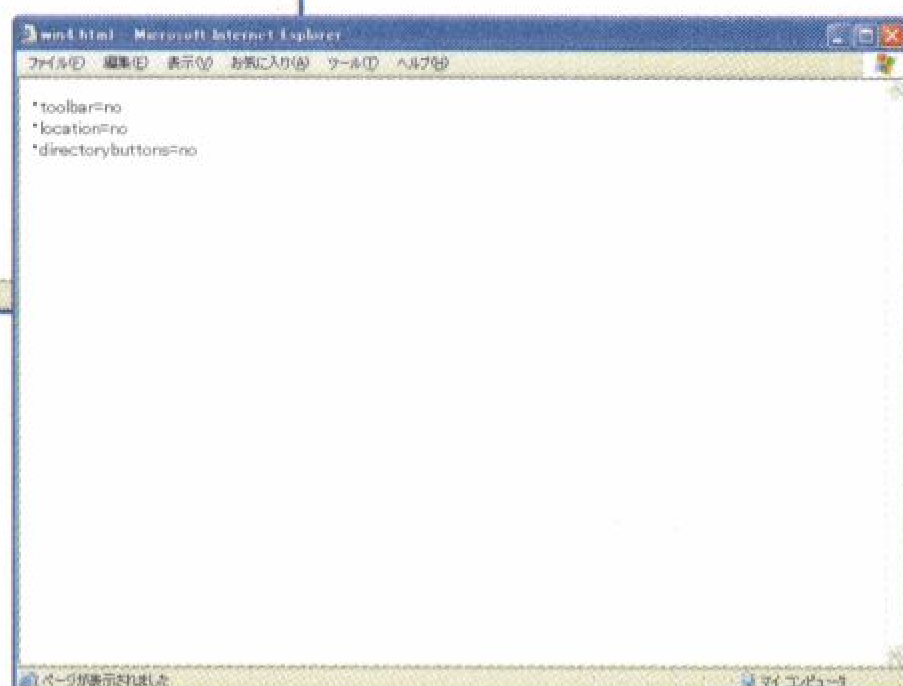
After

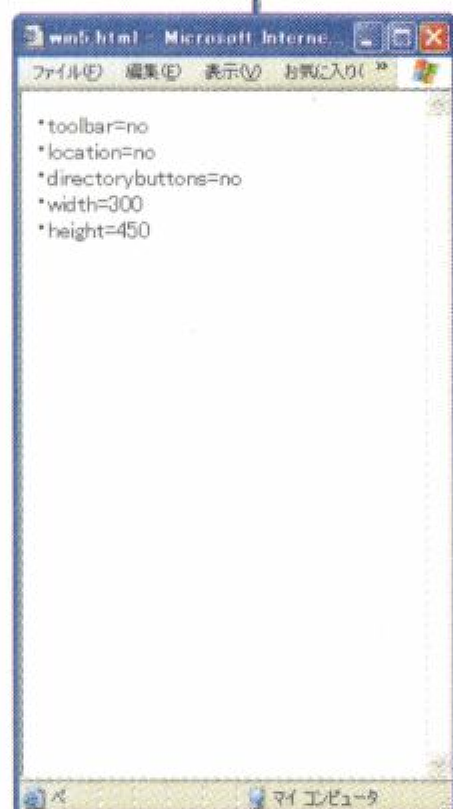








新しいウィンドウを開くには、「window.open("URL","ウィンドウ名","属性")」メソッドを使用します。
「URL」にはウィンドウ内に表示するHTMLファイルのURLを、「ウィンドウ名」には任意のウィンドウ名を指定します。ウィンドウ名を同じにすると、同じウィンドウとして取り扱われます。
「属性」の部分では、ウィンドウのツールバーやメニューバーなどの有無、ウィンドウのサイズなどを指定します。各項目は「,」で区切り、その項目が必要であれば「=yes」または「=1」と指定し、不要であれば「=no」または「=0」と指定し、ウィンドウのサイズは「=ピクセル」と指定します。指定できる属性は、「toolbar(ツールバー)」「location(ロケーションバー)」「directories(ディレクトリバー)」「status(ステータスバー)」「menubar(メニューバー)」「scrollbars(スクロールバー)」「resizable(リサイズボックス)」「width=pixels(ウィンドウの横幅)」「height(ウィンドウの縦幅)」で、各項目は省略すると「yes」として判断します。
Macintosh版のNetscape Navigator 2.0などの一部ブラウザでは、URLが引けないものがあります。これらのブラウザも対象に入れたWebページを作る場合は、コラム「Netscape Navigator 2.Xで「window.open()」を使用する時の注意」(次ページ)を参照してください。
実際に試す場合には、この他にも「win1.html」～「win5.html」の5つのHTMLファイルを用意してください。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function wopen1(){ window.open("win1.html","WindowOpen1",
                   "toolbar=yes,location=yes,directories=yes,statu
s=yes,menubar=yes,scrollbars=yes,resizable=yes") }
function wopen2(){ window.open("win2.html","WindowOpen2",
                   "toolbar=yes,location=yes,directories=no,status
=yes,menubar=yes,scrollbars=yes,resizable=yes") }
function wopen3(){ window.open("win3.html","WindowOpen3",
                   "toolbar=yes,location=no,directories=no,status=
yes,menubar=yes,scrollbars=yes,resizable=yes") }
function wopen4(){ window.open("win4.html","WindowOpen4",
                   "toolbar=no,location=no,directories=no,status=y
es,menubar=yes,scrollbars=yes,resizable=yes") }
function wopen5(){ window.open("win5.html","WindowOpen5",
                    "toolbar=no,location=no,directories=no,status=
yes,menubar=yes,scrollbars=yes,resizable=yes,width=300,height=450
") }
```

```
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*新しいウィンドウを開く
<p>
<form>
    <input type="button" value=" 全部あり " onClick="wopen1()"><br>
    <input type="button" value=" ディレクトリなし " onClick="wopen2()
"><br>
    <input type="button" value=" ディレクトリ・ロケーションなし " onClick
="wopen3()"><br>
    <input type="button" value=" ディレクトリ・ロケーション・ツールバーなし "
onClick="wopen4()"><br>
    <input type="button" value=" サイズ指定有り " onclick="wopen5()">
</form>
</p>
</body>
</html>
```

**注意**

## Netscape Navigator 2.Xで「window.open()」を使用する時の注意

　Machintosh版とUNIX版のNetscape Navigator 2.Xには、window.open()メソッドでURLが引けないという問題があります。そのため、通常のスクリプトの書き方では、開いたウィンドウに何も表示されません。

　もしも、それらのブラウザも対象としたページでwindow.open()を使用する場合は、

```
function wopen1(){ var W01;
                   W01=window.open(" ","WindowOpen6",
                   "toolbar=yes,location=yes,directories= yes,s
tatus=yes,menubar=yes,scrollbars=yes,resizable=yes");
                   W01.location.href="URL" }
```

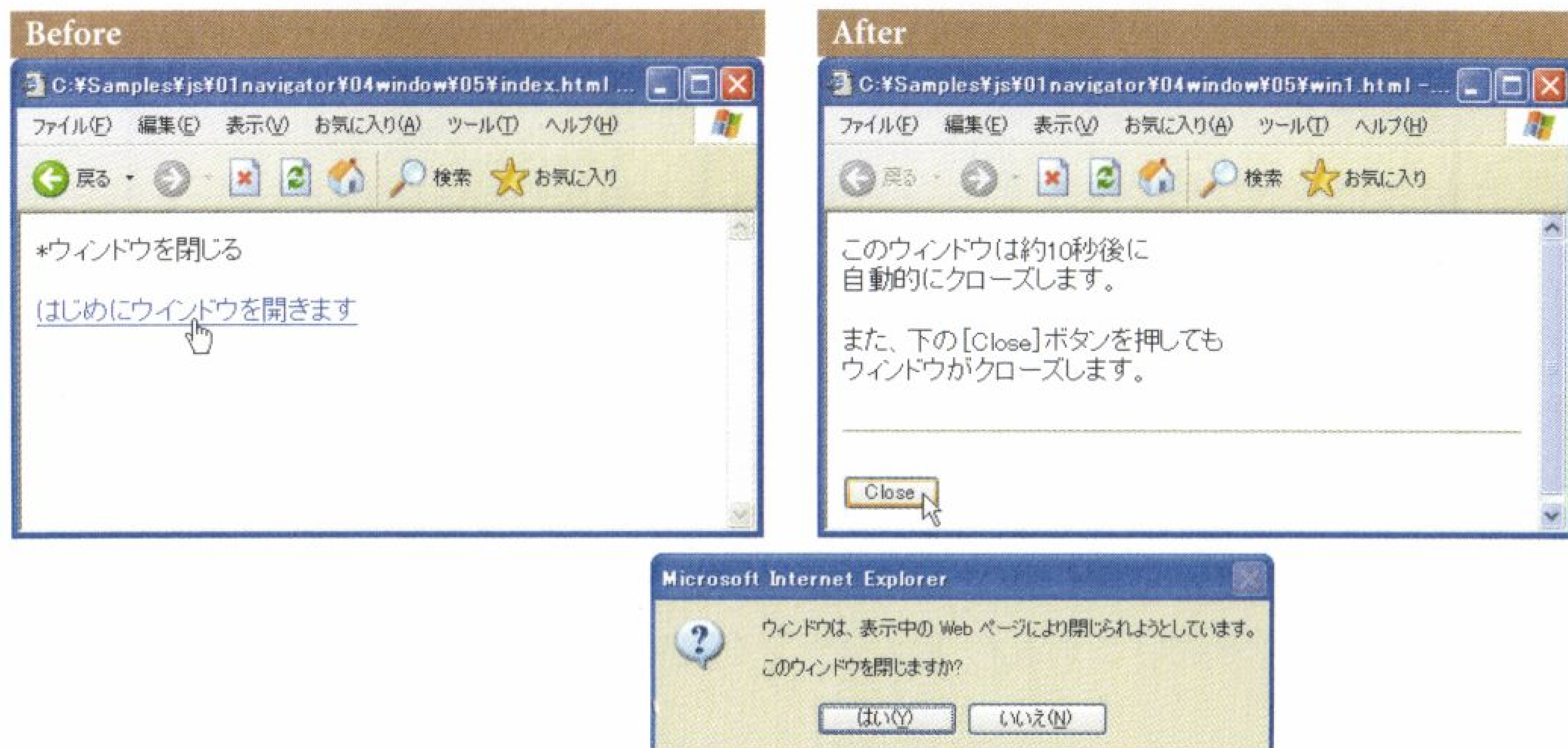
といったようにすれば大丈夫です。

　この時、URLは「http://」から始まるフルパスで指定してください。

　このスクリプトは、Macintosh版とUNIX版のNetscape Navigator 2.X以外でも正常に機能します。

# ウィンドウを閉じる

**window.close()** [メソッド]
**setTimeout(*処理,時間設定*)** [メソッド]
**onLoad="*スクリプト | 関数*"** [イベントハンドラ]




「close()」メソッドは、ウィンドウを閉じます。
サンプルでは、ページがロードされた時に<body>内のイベントハンドラ「onLoad」が「setTimeout()」メソッドを呼び出し、10秒後に「window.close()」が発生してウィンドウが閉じます。また、フォームのボタンを押しても、イベントハンドラ「onClick」が「window.close()」を発生させ、ウィンドウが閉じます。
setTimeoutはミリ秒(1000分の1秒)単位で時間を設定できますが、あまり小さい数字を設定しても効果は出ません。
Netscape Navigator 3.0からは、セキュリティ上の対策のため、「close()」メソッドが発生した時にウィンドウが来歴情報を持っていれば、閉じる時に確認のダイアログボックスが開くようになりました。

## Sample

**[win1.html]**

```
<body onLoad="setTimeout('window.close()',10000)">
このウィンドウは約10秒後に<br>自動的にクローズします。
<p>
また、下の[Close]ボタンを押しても<br>ウィンドウがクローズします。
</p>
<hr>
<form>
    <input type="button" value=" Close " onclick="window.close()">
</form>
</body>
```

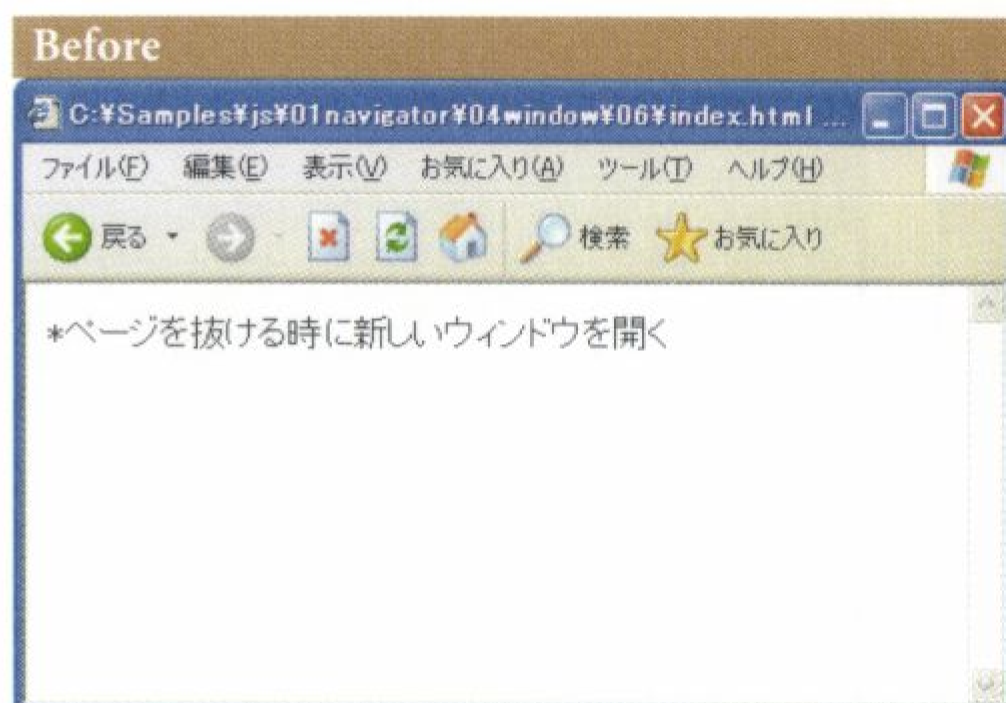
IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

JavaScript

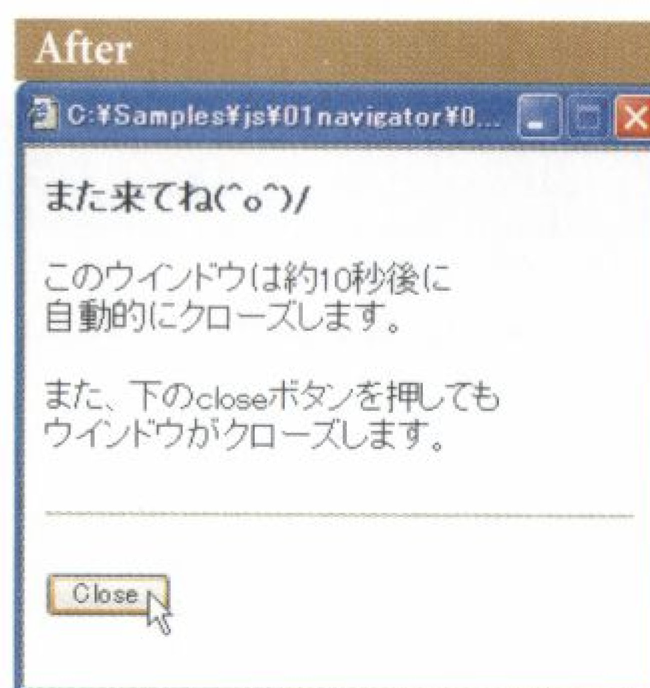
# ページを抜ける時に新しいウィンドウを開く

**window.open()** [メソッド]
**onUnload="*スクリプト | 関数*"** [イベントハンドラ]

Before




After




サンプルでは、別のページに抜ける時に、イベントハンドラ「onUnload」が関数を発生させてウィンドウを開き、お礼のメッセージを書いたHTMLファイルをロードしています。

## Sample

```
<html><head>
<title></title>
<script type="text/javascript">
<!--
function wopen(){ window.open("bay.html","WindowOpen1",
                  "toolbar=no,location=no,directories=no,width=300
,height=250");
}
//-->
</script>
  ～中略～
</head>
<body onUnload="wopen()">
*ページを抜ける時に新しいウィンドウを開く
</body></html>
```

**注意**

### onUnloadを使用する時の注意

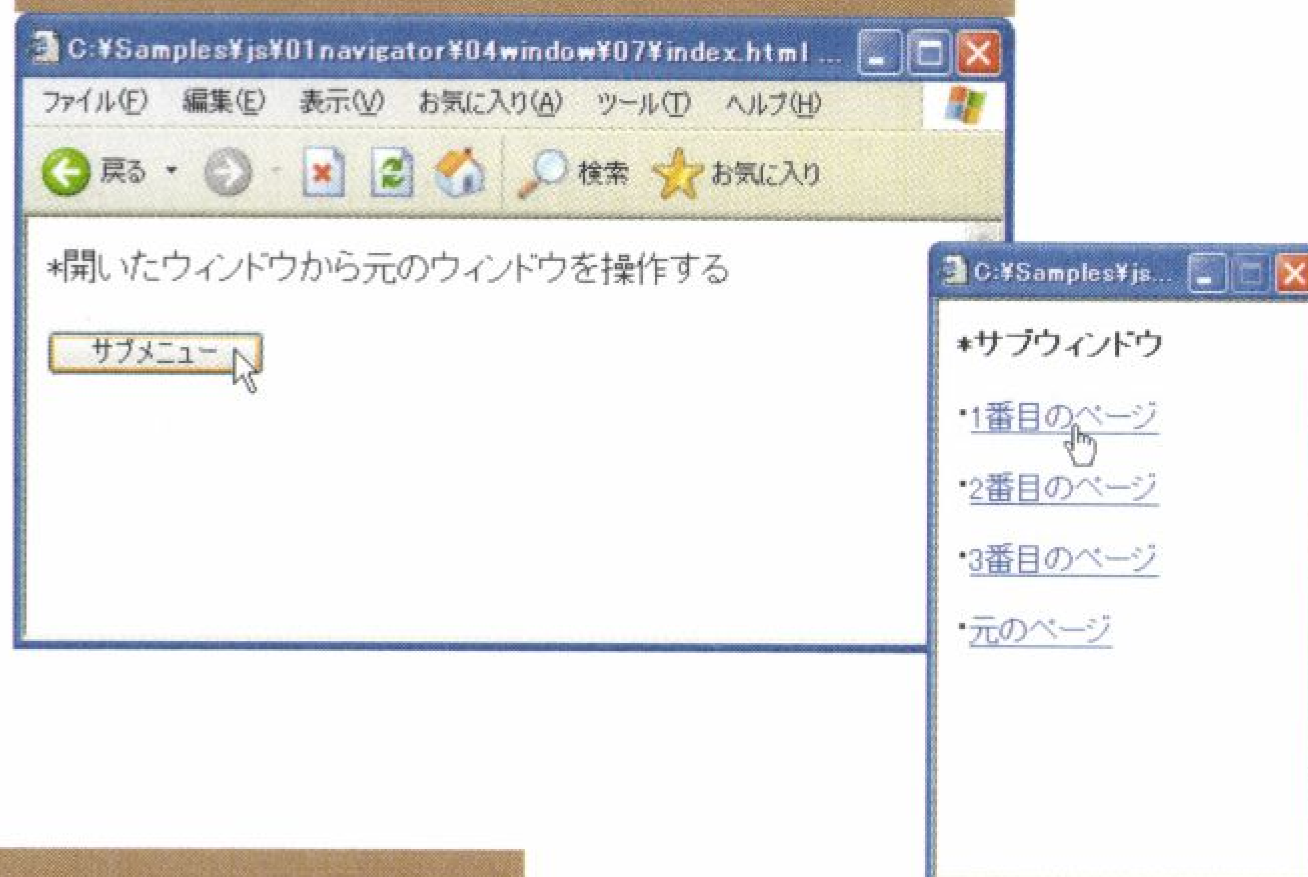
イベントハンドラ「onUnload」を使って複雑な処理をさせると、OSやバージョンに関係なく、ブラウザの動作が不安定になる場合があります。
「onUnload」を使用する時は、十分なテストの上で使用してください。

# 開いたウィンドウから元のウィンドウを操作する

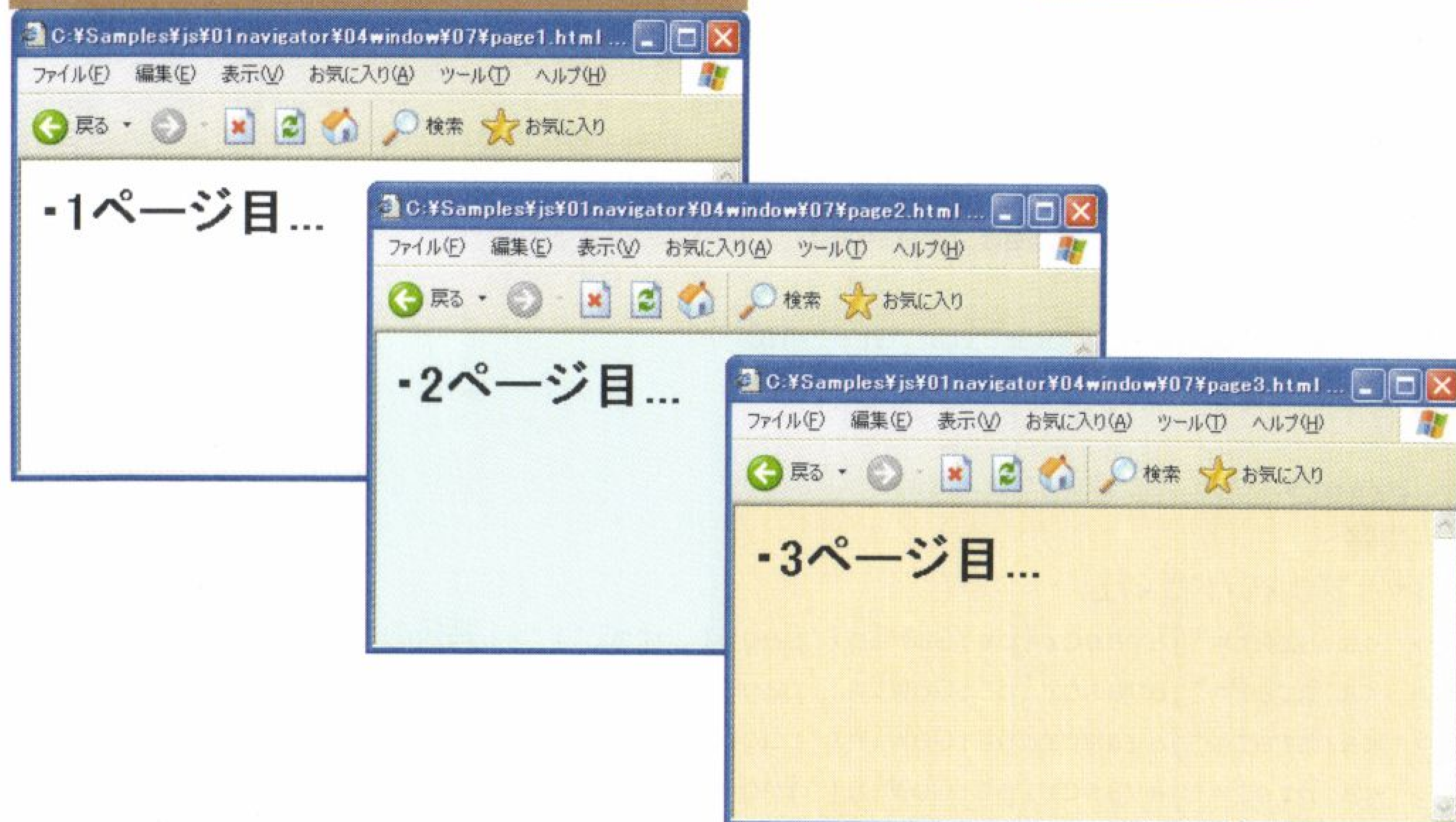
**opener** [プロパティ]
**closed** [プロパティ]



Before




After




「opener」プロパティは、元のウィンドウを参照します。
サンプルでは、サブウィンドウ内のリンクがクリックされると、「opener.location.href」でサブウィンドウをオープンしたメインウィンドウが参照され、メインウィンドウに「GoWin()」内で指定しているURLがロードされます。
また、リンクがクリックされた時にメインウィンドウが閉じられていても、「closed」プロパティでウィンドウが閉じられている状態を取得しますので、「opener.closed」が真(true)となって新たにウィンドウが開き、そこにページが読み込まれます。
JavaScript1.1で追加されたメソッドです。
バージョン5.X以前のMacintosh版など、一部のInternet Explorerでは、「closed」プロパティがうまく機能しない場合があります。

## Sample

**【メインウィンドウ】**

```
<script type="text/javascript">
<!--
function SWwopen(){
        window.open("menu.html","SWindow", "toolbar=no,location=no,directories=no,status=no,menubar=no,scrollbars=no,resizable=no,width=200,height=300");
    }
//-->
</script>
  ～中略～
<p>
<form>
    <input type="button" value=" サブメニュー " onClick="SWwopen()">
</form>
</p>
```

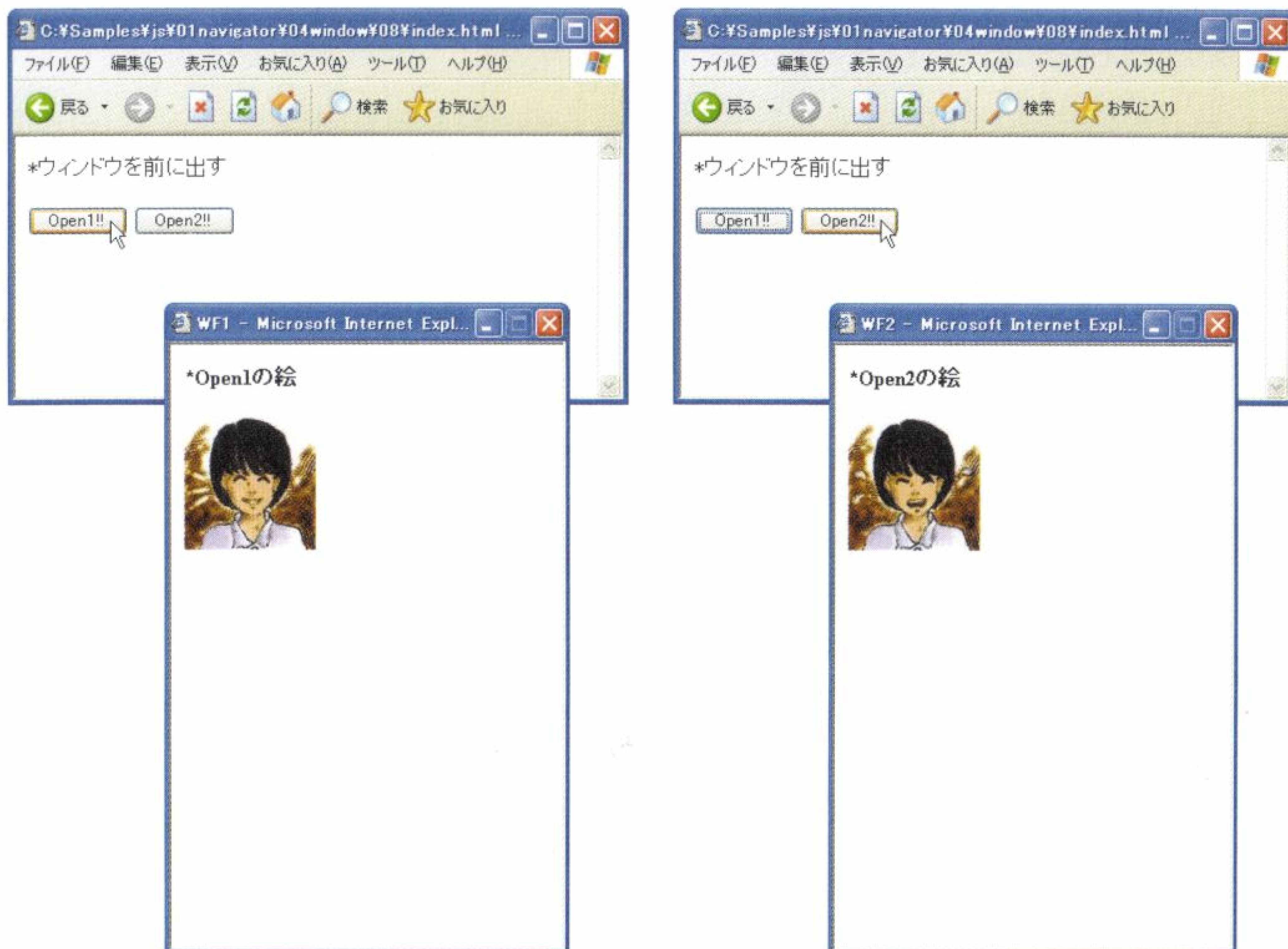
**【サブウィンドウ(menu.html)】**

```
<script type="text/javascript">
<!--
function GoWin(WO) {
    if (opener.closed) {
        NewWin=window.open("","MWindow");
        NewWin.location.href=WO;
                                }
    else { opener.location.href=WO }
}
//-->
</script>
  ～中略～
<b>* サブウィンドウ</b>
<p>・<a href="javascript:GoWin('page1.html')">1番目のページ</a></p>
<p>・<a href="javascript:GoWin('page2.html')">2番目のページ</a></p>
<p>・<a href="javascript:GoWin('page3.html')">3番目のページ</a></p>
<p>・<a href="javascript:GoWin('index.html')">元のページ</a></p>
```

# ウィンドウを前に出す

## window.focus() [メソッド]






「focus()」メソッドは、ウィンドウにフォーカスを与えます。
サンプルでは、ウィンドウ新しく開いたり、新しく開いたウィンドウが書き変わった時に、もしそのウィンドウが他のウィンドウの後ろに隠れていても、1番手前に移動します。
JavaScript1.1で追加されたメソッドです。

### Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function wopen1(){
    var WO1;
    WO1=window.open("","WindowFocus1", "toolbar=no,location=no,dir
ectories=no,width=300,height=450");
```

```
    WO1.focus();
    WO1.document.write("<html><title>WF1</title></head>");
    WO1.document.write("<body>");
    WO1.document.write("<b>*Open1の絵</b>");
    WO1.document.write("<p>");
    WO1.document.write("<img name='image1' src='image1.jpg' alt='I
MAGE1'>");
    WO1.document.write("</body>");
    WO1.document.write("</html>");
    WO1.document.close();
}
function wopen2(){
    var WO1;
    WO1=window.open("","WindowFocus1", "toolbar=no,location=no,dir
ectories=no,width=300,height=450");
    WO1.focus();
    WO1.document.write("<html><title>WF2</title></head>");
    WO1.document.write("<body>");
    WO1.document.write("<b>*Open2の絵</b>");
    WO1.document.write("<p>");
    WO1.document.write("<img name='image2' src='image2.jpg' alt='i
mage2'>");
    WO1.document.write("</body>");
    WO1.document.write("</html>");
    WO1.document.close();
}
//-->
</script>
</head>
<body>
*ウィンドウを前に出す
<p>
<form>
    <input type="button" value=" Open1!! " onClick="wopen1()">
    <input type="button" value=" Open2!! " onClick="wopen2()">
</form>
</p>
</body></html>
```



**注意**

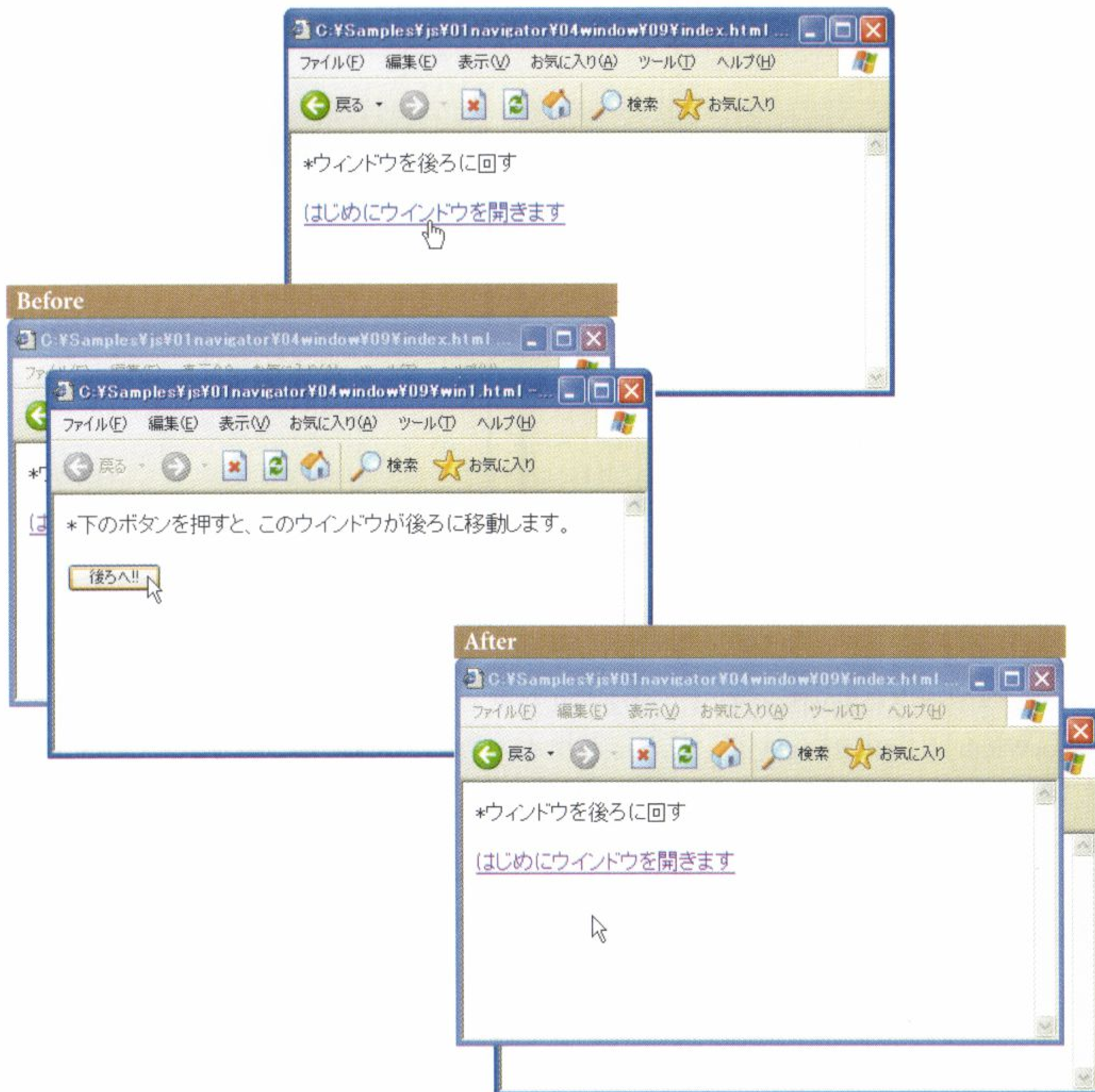
## 小さなウィンドウを開く時の注意

「window.open()」のウィンドウサイズ指定で、ウィンドウの横幅がステータスバー内のファイルの読み込み状況を示すバーよりも小さく指定されていると、Macintosh PPC版Netscape Navigator 3.0ではシステムエラーを起こすことがあります。

# ウィンドウを後ろに回す

## window.blur() [メソッド]






「blur()」メソッドは、ウィンドウからフォーカスを移動させます。
サンプルでは、複数のウィンドウが開いている時にボタンが押されると「window.blur()」が評価され、ウィンドウが後ろに回ります。
JavaScript1.1で追加されたメソッドです。
Netscape 6.Xでは、このスクリプトは正常に動作しません。

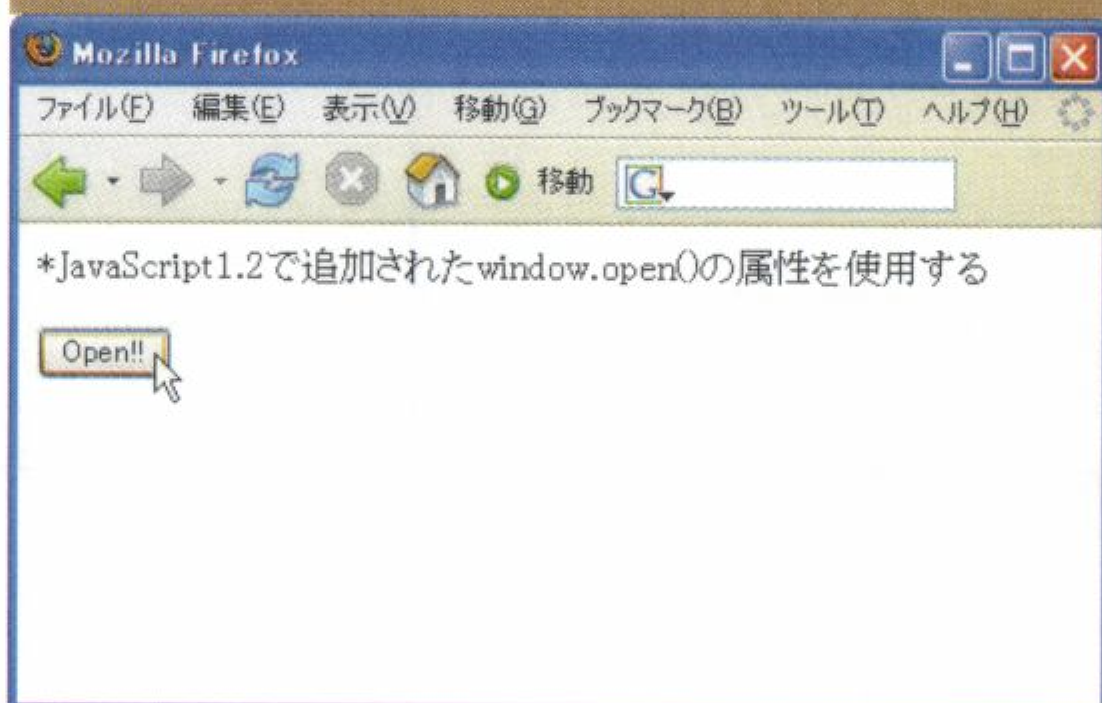
### Sample

```
<form>
    <input type="button" value=" 後ろへ!! " onClick="window.blur()">
</form>
```

# JavaScript1.2で追加された window.open()の属性を使用する

**window.open("*URL*","*ウィンドウ名*","*属性*")** [メソッド]

Before




After




「open()」メソッドにJavaScript1.2で追加されたウィンドウ属性は、次の通りです。「outerWidth」と「outerHeight」は、それぞれウィンドウの外側の幅と高さを、ピクセル単位で指定します。外側のサイズなので、ツールバーやロケーションボックスなどがあってもなくても、ウィンドウのサイズは変わりません。「left」と「top」は、新しく開くウィンドウの左上角の位置を、ディスプレイの左上角の上からと左からのピクセル単位で指定します。
サンプルでは、ボタンをクリックしたタイミングで、ディスプレイの左上角から左へ200ピクセルで下へ200ピクセルの位置に左上角がくるように、ツールバーも含めて幅400ピクセルで高さ400ピクセルのウィンドウを開いています。
Internet Explorerでは、「outerWidth」「outerHeight」はサポートされていません。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function WO10_0(){
    var WO;
    WO=window.open("","Win4","toolbar=yes,location=no,directories=
no,outerWidth=400,outerHeight=400,left=200,top=200");
```

```
    WO.document.open();
    WO.document.write("<html><title>Win4.0</title>");
    WO.document.write("</head>");
    WO.document.write("<body>");
    WO.document.write("・ウィンドウの外周: 400x400(ピクセル)");
    WO.document.write("<br>");
    WO.document.write("・ディスプレイの左上角からの表¥示¥位置: 200X200(ピクセ
ル)");
    WO.document.write("</body>");
    WO.document.write("</html>");
    WO.document.close();
}
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*JavaScript1.2で追加されたwindow.open()の属性を使用する
<p>
<form>
    <input type="button" value=" Open!! " onClick="WO10_0()">
</form>
</p>
</body></html>
```

**注意**

## open()の属性を使用する時の注意

JavaScript1.2で追加されたopen()の属性のうち、「alwaysLowered」「alwaysRaised」「z-lock」は、「Signed Script」内で設定する必要があります。

また、「innerWidth」「innerHeight」「outerWidth」「outerHeight」で100×100ピクセル以下にウィンドウサイズを指定する場合や、「screenX」「screenY」でディスプレイの表示領域外にウィンドウ表示位置を指定する時、「titlebar」の指定でタイトルバーを表示しないように指定する時にも、「Signed Script」内で設定する必要があります。

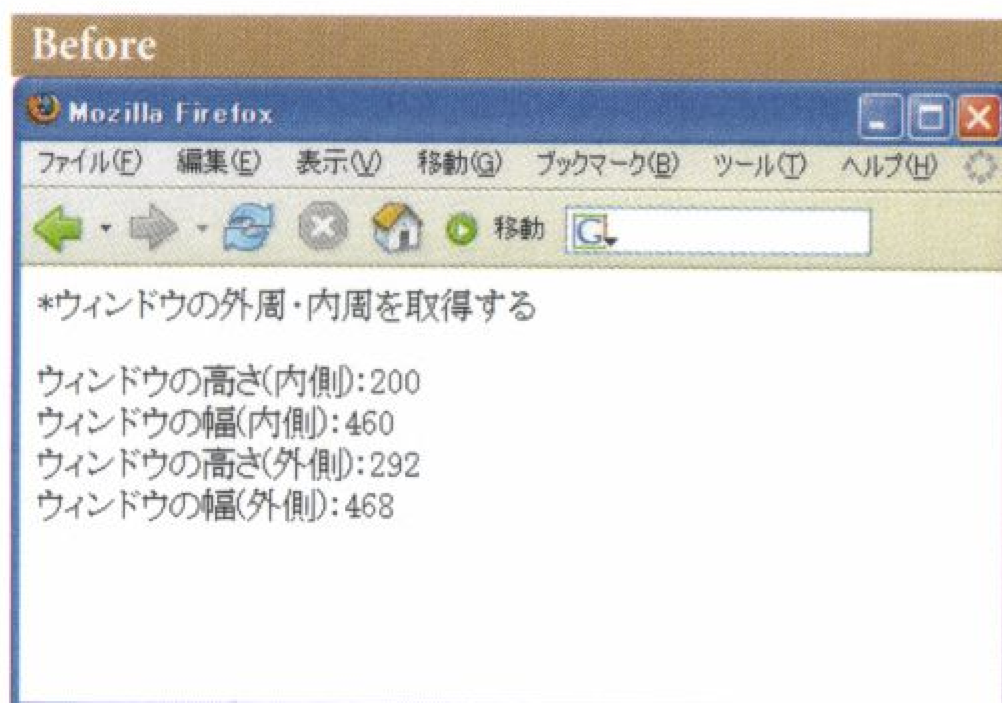
「alwaysLowered」「alwaysRaised」「z-lock」は、それを表示するプラットホームによって動きが異なる場合があります。たとえば「z-lock」の場合、Windows 95では他のアプリケーションのウィンドウが開いていても、それらより後ろに新しいウィンドウが開きます。しかし、Macintoshの場合、他のアプリケーションのウィンドウがあると、そのウィンドウより前に新しいウィンドウが開く場合があります。

# ウィンドウの外周・内周を取得する

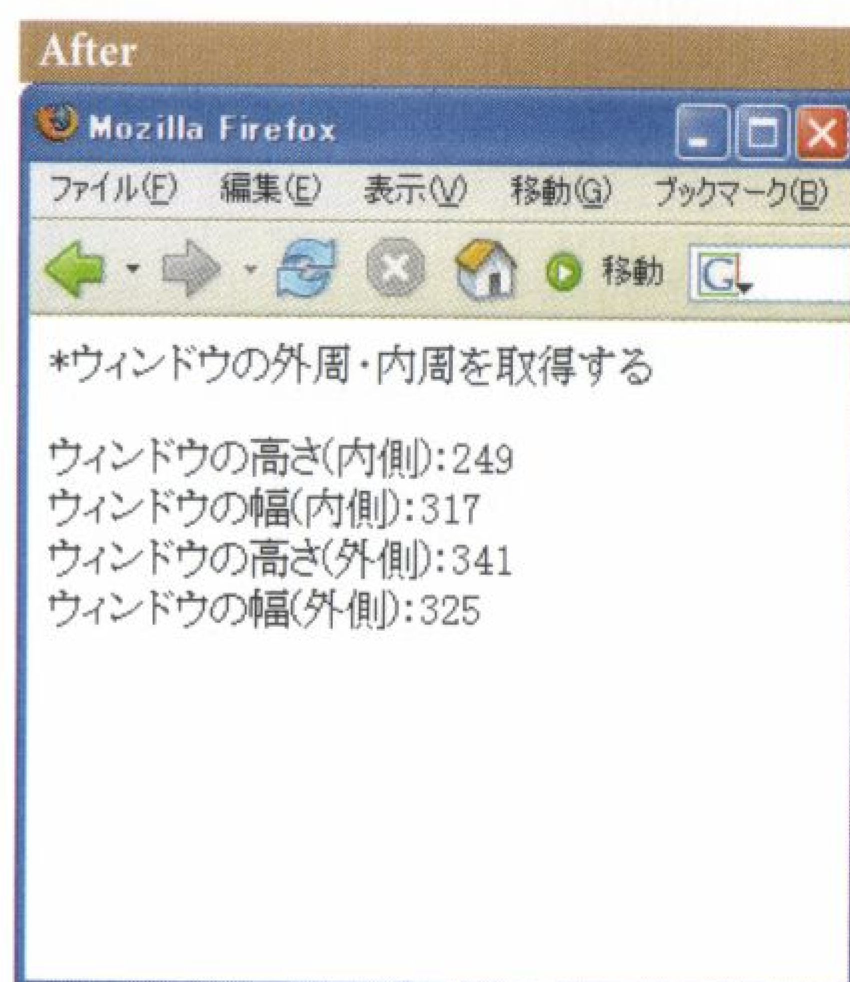
| | |
|---|---|
| **window.innerHeight** | [プロパティ] |
| **window.innerWidth** | [プロパティ] |
| **window.outerHeight** | [プロパティ] |
| **window.outerWidth** | [プロパティ] |



Before




After




「innerHeight」プロパティと「innerWidth」プロパティはウィンドウ内の表示領域の高さと幅の値を、「outerHeight」プロパティと「outerWidth」プロパティはツールバーやステータスバーなども含めたウィンドウの外側の高さと幅の値を、それぞれ持っています。
これらのプロパティは、Internet Explorer ではサポートされていません。
JavaScript1.2 で追加されたプロパティです。

## Sample

```
<script type="text/javascript">
<!--
    document.write("ウィンドウの高さ(内側):",window.innerHeight);
    document.write("<br>");
    document.write("ウィンドウの幅(内側):",window.innerWidth);
    document.write("<br>");
    document.write("ウィンドウの高さ(外側):",window.outerHeight);
    document.write("<br>");
    document.write("ウィンドウの幅(外側):",window.outerWidth);
//-->
</script>
```

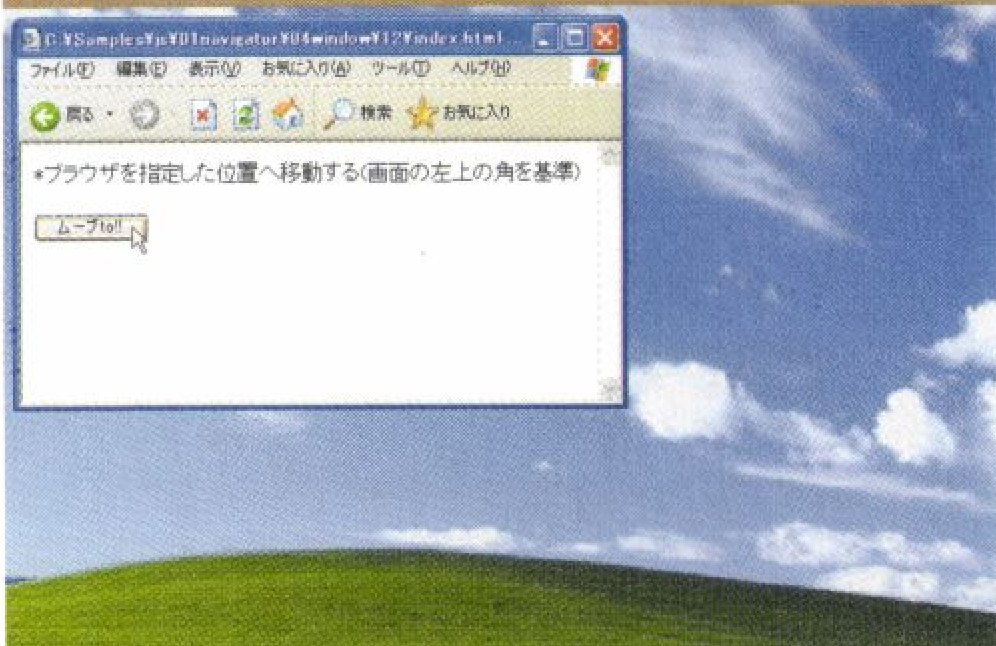
# ブラウザを指定した位置へ移動する

**window.moveTo(x,y)** [メソッド]

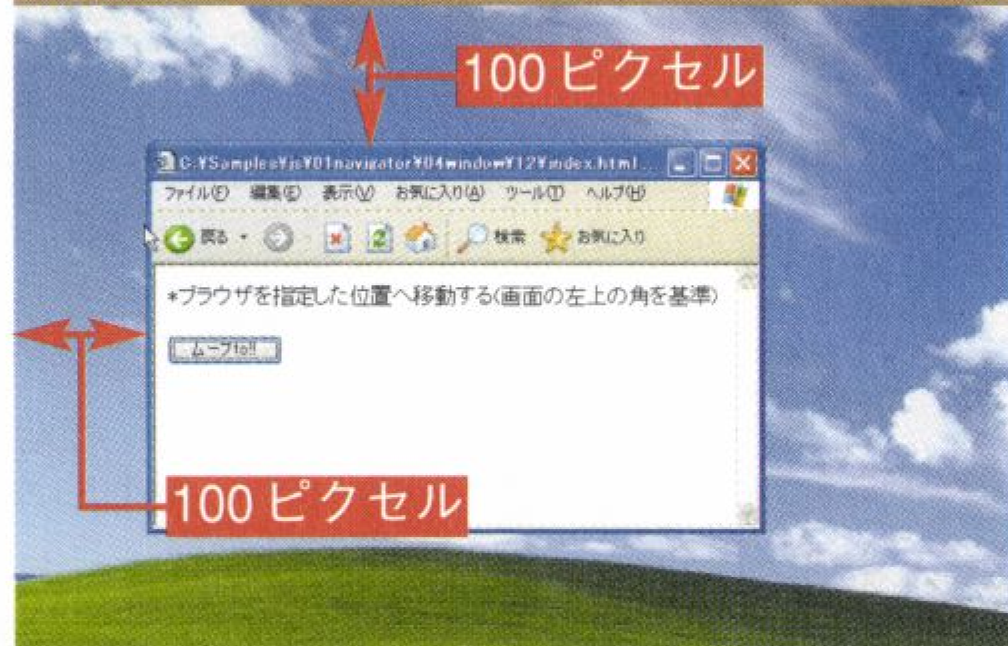
| | |
|---|---|
| x | ディスプレイ左上から下方向(ピクセル) |
| y | ディスプレイ左上から右方向(ピクセル) |



Before




After




「moveTo()」メソッドは、ピクセル単位で指定した位置(x,y)へ、ウィンドウを移動させます。
サンプルでは、ウィンドウがどこの位置に表示されていても、ボタンをクリックするとディスプレイの左上角から下へ100 ピクセルで右へ100 ピクセルの位置に、ウィンドウの左上角がくるように移動します。
JavaScript1.2 で追加されたメソッドです。



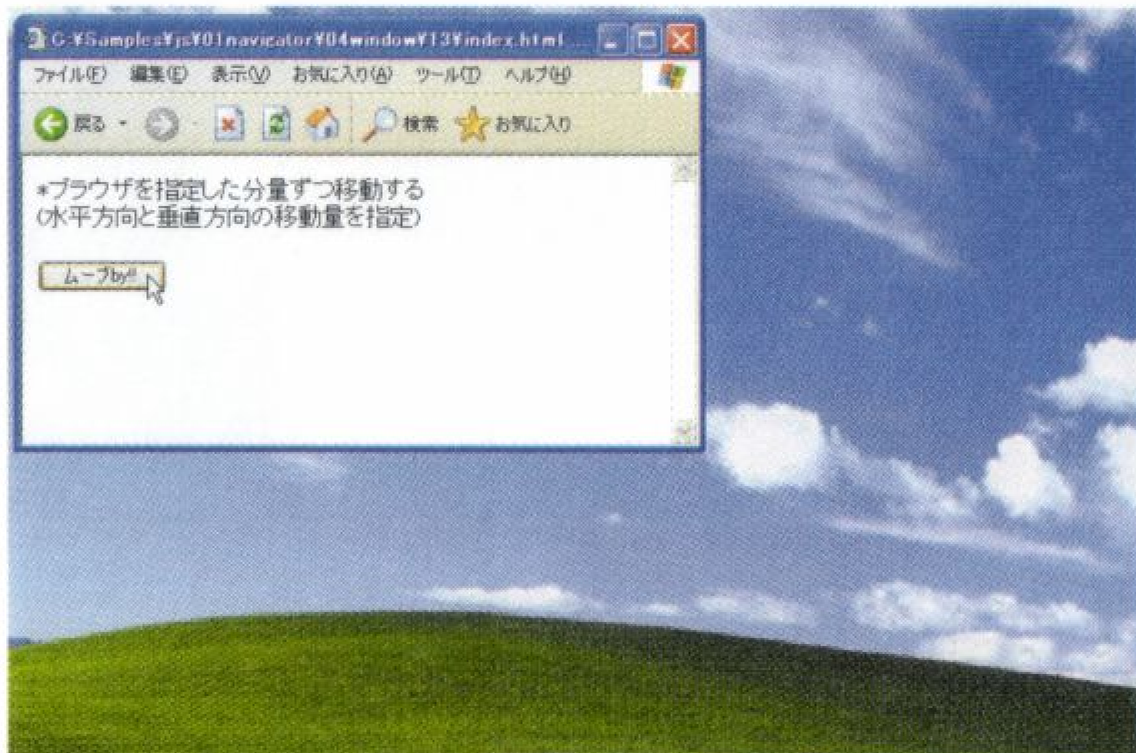
## Sample

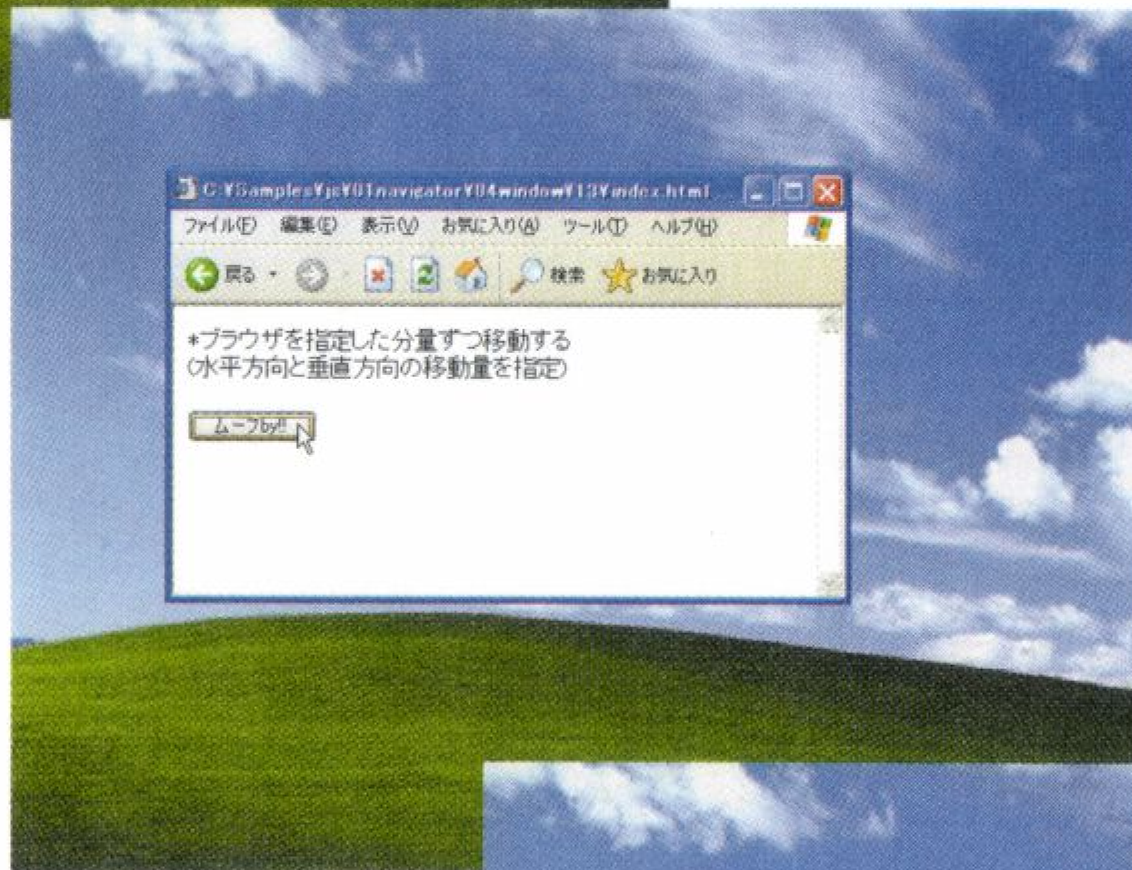
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function MVto(){ window.moveTo(100,100) }
//-->
</script>
</head>
<body>
*ブラウザを指定した位置へ移動する(画面の左上の角を基準)
<p>
<form>
    <input type="button" value=" ムーブto!! " onClick="MVto()">
</form>
</p>
</body>
</html>
```

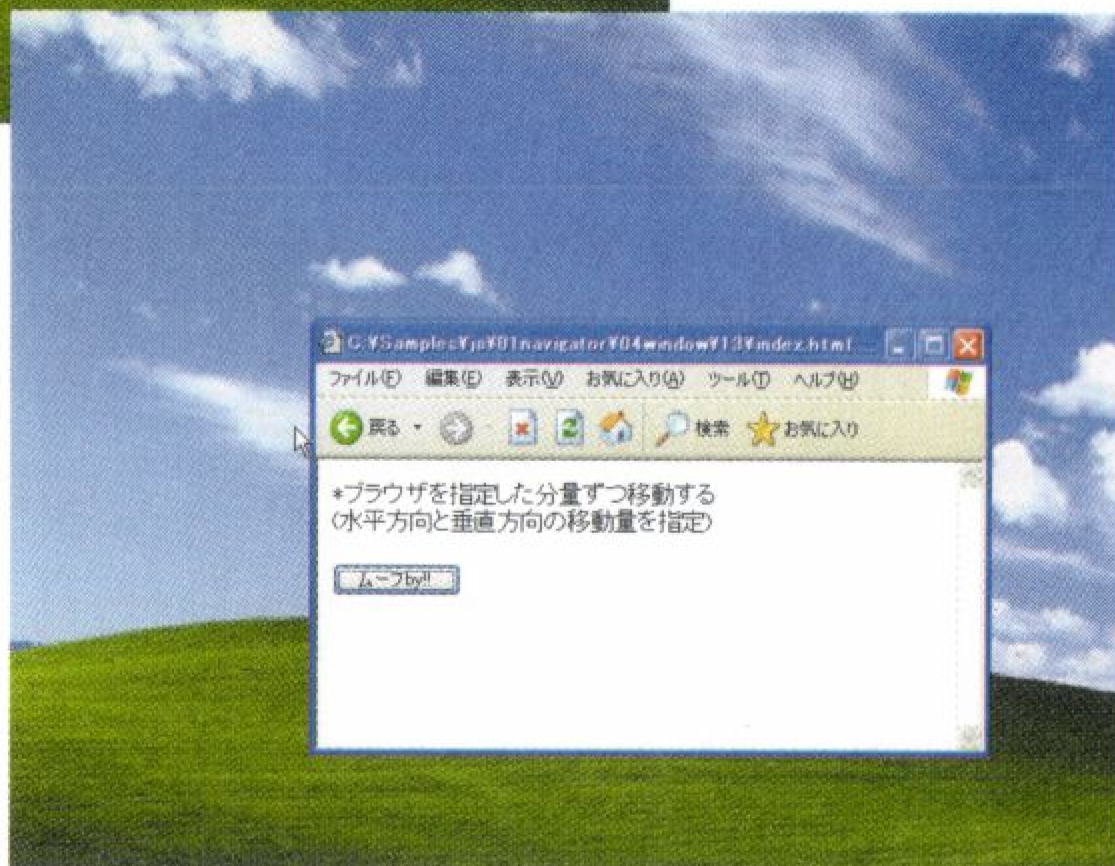
# ブラウザを指定した分量ずつ移動する

| | |
|---|---|
| **window.moveBy(*x*,*y*)** | [メソッド] |
| x | 水平方向への移動量(ピクセル) |
| y | 垂直方向への移動量(ピクセル) |






「moveBy()」メソッドは、ピクセル単位でウィンドウを移動させます。
サンプルでは、ボタンをクリックするごとに、ウィンドウが右と下へ100ピクセルずつ移動します。
JavaScript1.2で追加されたメソッドです。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function MVby(){ window.moveBy(100,100) }
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*ブラウザを指定した分量ずつ移動する<br>
（水平方向と垂直方向の移動量を指定）
<p>
<form>
    <input type="button" value=" ムーブby!! " onClick="MVby()">
</form>
</p>
</body>
</html>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Safari IE5-mac IE4-mac

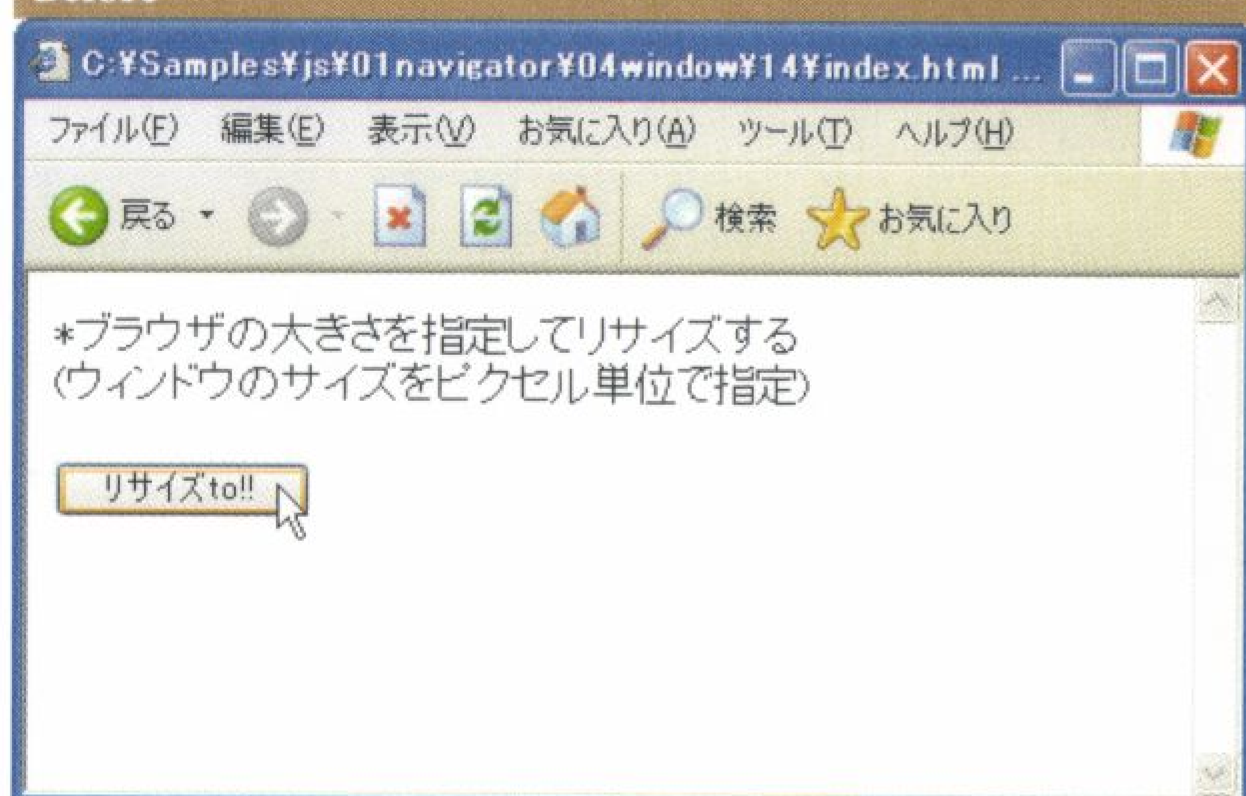
ウィンドウを操作する

JavaScript

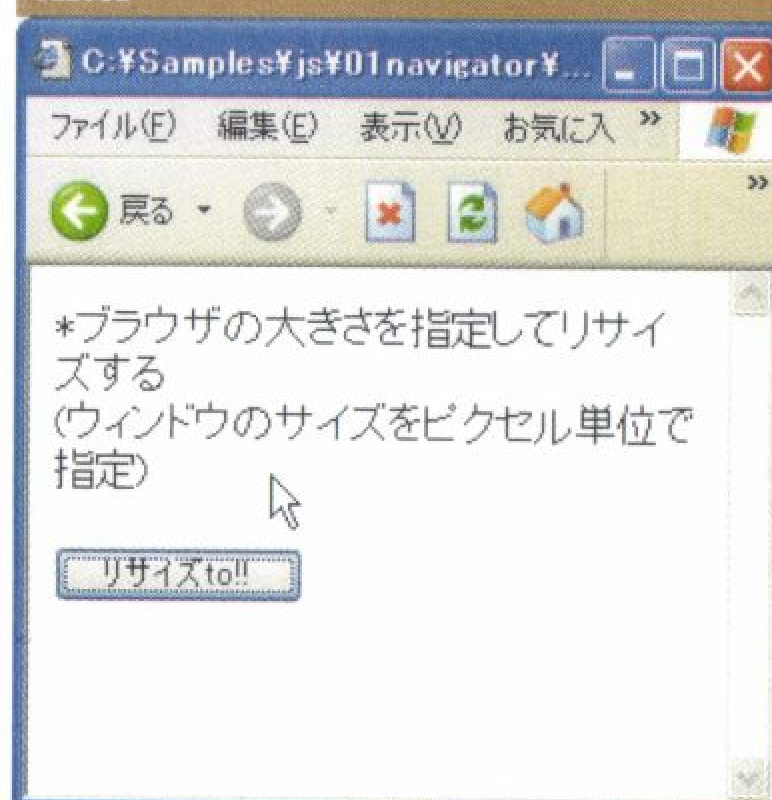
# ブラウザの大きさを指定してリサイズする

| window.resizeTo(x,y) | [メソッド] |
|---|---|
| x | ウィンドウ左上角から垂直方向(ピクセル) |
| y | ウィンドウ左上角から水平方向(ピクセル) |

Before




After




「resizeTo()」メソッドは、ピクセル単位で指定したサイズにウィンドウをリサイズします。サンプルでは、ボタンをクリックすると、ウィンドウの左上角を基準に、縦横300ピクセルにリサイズします。
なお、設定したサイズは、Internet Explorerではウィンドウの外周のサイズ、Netscape Navigatorではウィンドウ内の表示領域になるため、Internet ExplorerとNetscape Navigatorでは、スクリプトの実行結果が異なります。
JavaScript1.2で追加されたメソッドです。

## Sample

```
<script type="text/javascript">
<!--
function RSto(){ window.resizeTo(300,300) }
//-->
</script>
 ～中略～
<body>
*ブラウザの大きさを指定してリサイズする<br>
 (ウィンドウのサイズをピクセル単位で指定)
<p>
<form>
    <input type="button" value=" リサイズto!! " onClick="RSto()">
</form>
</p>
</body>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

ウィンドウを操作する

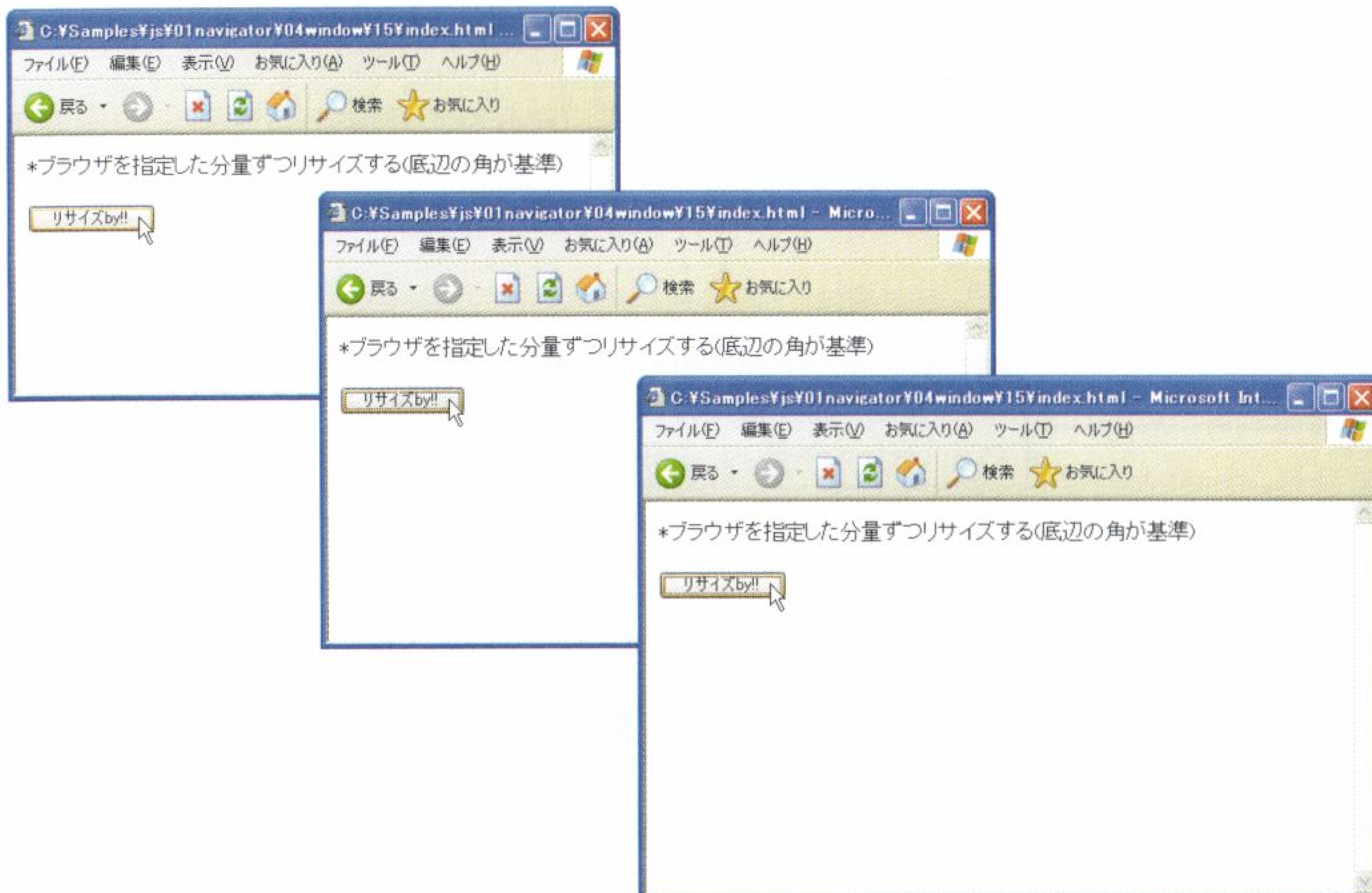
# ブラウザを指定した分量ずつリサイズする

**window.resizeBy(*x*,*y*)** [メソッド]

**x** 底辺のリサイズ量（ピクセル）

**y** 右辺のリサイズ量(ピクセル)








「resizeBy()」メソッドは、ウィンドウのサイズをピクセル単位で変更します。
サンプルでは、ウィンドウの底辺と右辺が、ボタンをクリックするごとにそれぞれ50ピクセルずつ広がります。
JavaScript1.2で追加されたメソッドです。

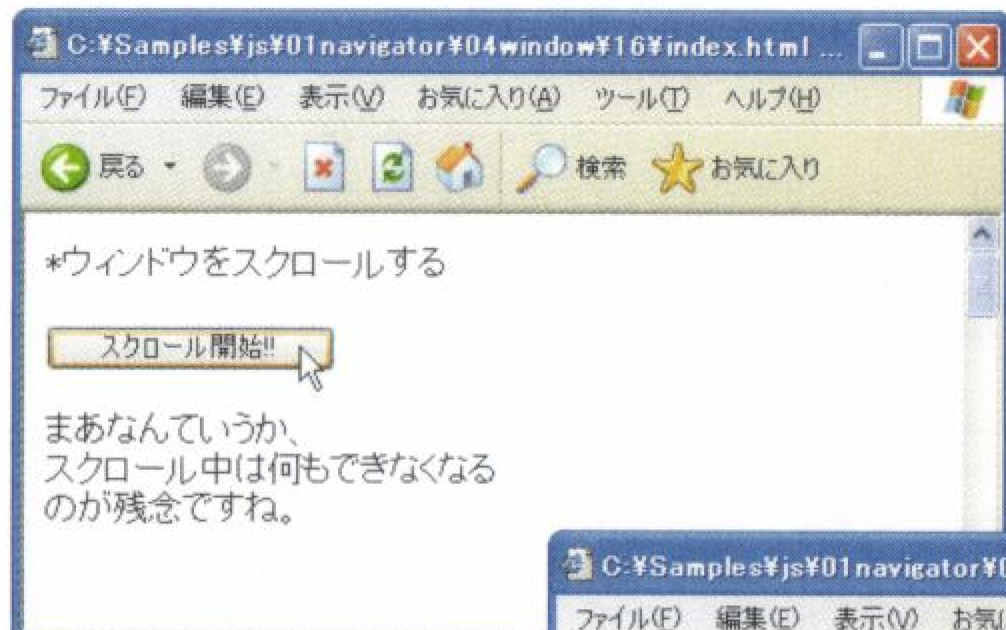
## Sample

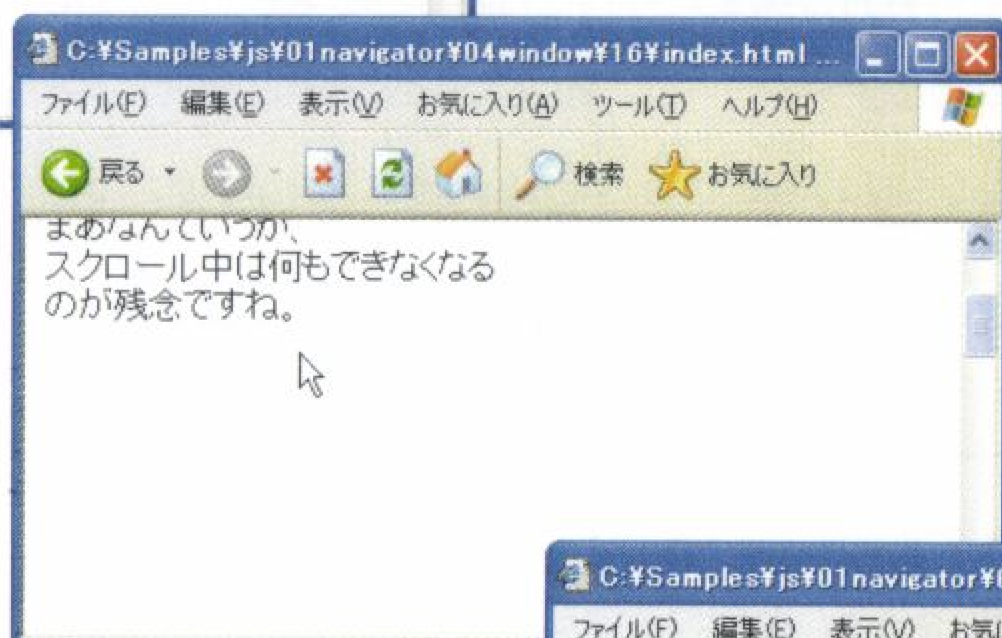
```
<script type="text/javascript">
<!--
function RSby(){ window.resizeBy(50,50) }
//-->
</script>
  ～中略～
<p>
<form>
<input type="button" value=" リサイズby!! " onClick="RSby()">
</form>
</p>
```

# ウィンドウをスクロールする

**window.scroll(*x*,*y*)** [メソッド]

| | |
|---|---|
| x | ウィンドウ左上角から垂直方向(ピクセル) |
| y | ウィンドウ左上角から水平方向(ピクセル) |







「scroll(x,y)」メソッドは、x軸とy軸をピクセル単位で指定し、指定した位置までブラウザをスクロールさせます。
サンプルでは、for文でy軸に数値を代入し続け、ブラウザを縦方向にスクロールさせています。
for文で式をループさせているだけなので、実行速度はマシンパワーに左右され、実行中は他の動作を受け付けなくなります。
JavaScript1.1で追加されたメソッドですが、Netscape Navigator 2.0でも動作します。
このスクリプトは、Mac OS 9版のInternet Explorer 5.Xでは動作しますが、Mac OS X版では動作しません。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function SCLL() {
    for(var i=0; i<150;i++) { window.scroll(0,i) }
    for(i=149; i>=0;i--) { window.scroll(0,i) }
}
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
＊ウィンドウをスクロールする
<p>
<form>
    <input type="button" value=" スクロール開始!! " onClick="SCLL()">
</form>
</p>
まあなんていうか、<br>
スクロール中は何もできなくなる<br>
のが残念ですね。
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
ダミーです...。
</body>
</html>
```

**注意**

## 「scroll()」メソッドを使用する時の注意

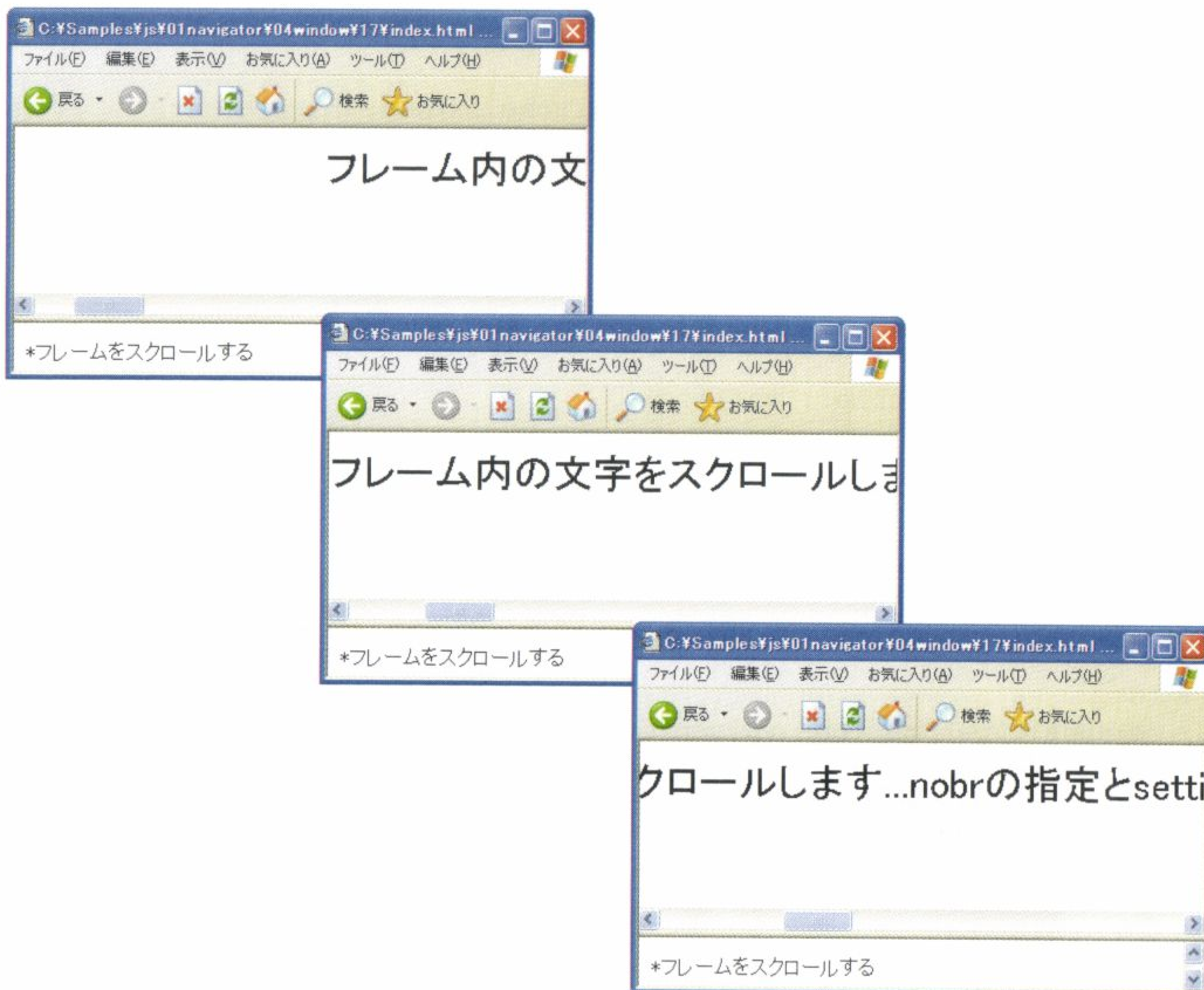
Windows版のNetscape Navigatorでは、「scroll(x,y)」でスクロール位置を指定していても、その範囲にドキュメントがなければドキュメントがある範囲しかスクロールしません。

また、フレームや「window.open()」の指定でスクロールバーが出ないように指定していてもスクロールしないので、注意してください。

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# フレームをスクロールする

| | |
|---|---|
| **window.scroll(*x*,*y*)** | [メソッド] |
| **x** | フレーム左上角から垂直方向(ピクセル) |
| **y** | フレーム左上角から水平方向(ピクセル) |




サンプルでは、フレーム内のウィンドウを、「scroll(x,y)」メソッドのx軸の値に「setTimeout()」メソッドで一定時間ごとにiの値を1ずつ加えながら代入することによって、横にスクロールさせています。

一定時間に処理を繰り返しているので、実行中は他の動作を受け付けなくなるようなことはありません。

スクロールさせる文字は、<nobr>を使って改行されないようにしています。

JavaScript1.1で追加されたメソッドですが、Netscape Navigator 2.0でも動作します。

## Sample

**【フレームウィンドウ】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" >
<html>
<head>
```

```
<title></title>
</head>
<frameset rows="150,*">
    <frame src="f1.html" name="f1">
    <frame src="f2.html" name="f2">
</frameset>
<noframes>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</noframes>
</html>
```

【f1.html】

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
var i = 0;
var MOZI = 3200;
var ST = 50;
function SCLL2() {
    if ( i < MOZI) { i = i+2;
           window.scroll(i,0);
           }
             setTimeout("SCLL2()", ST);
}
//-->
</script>

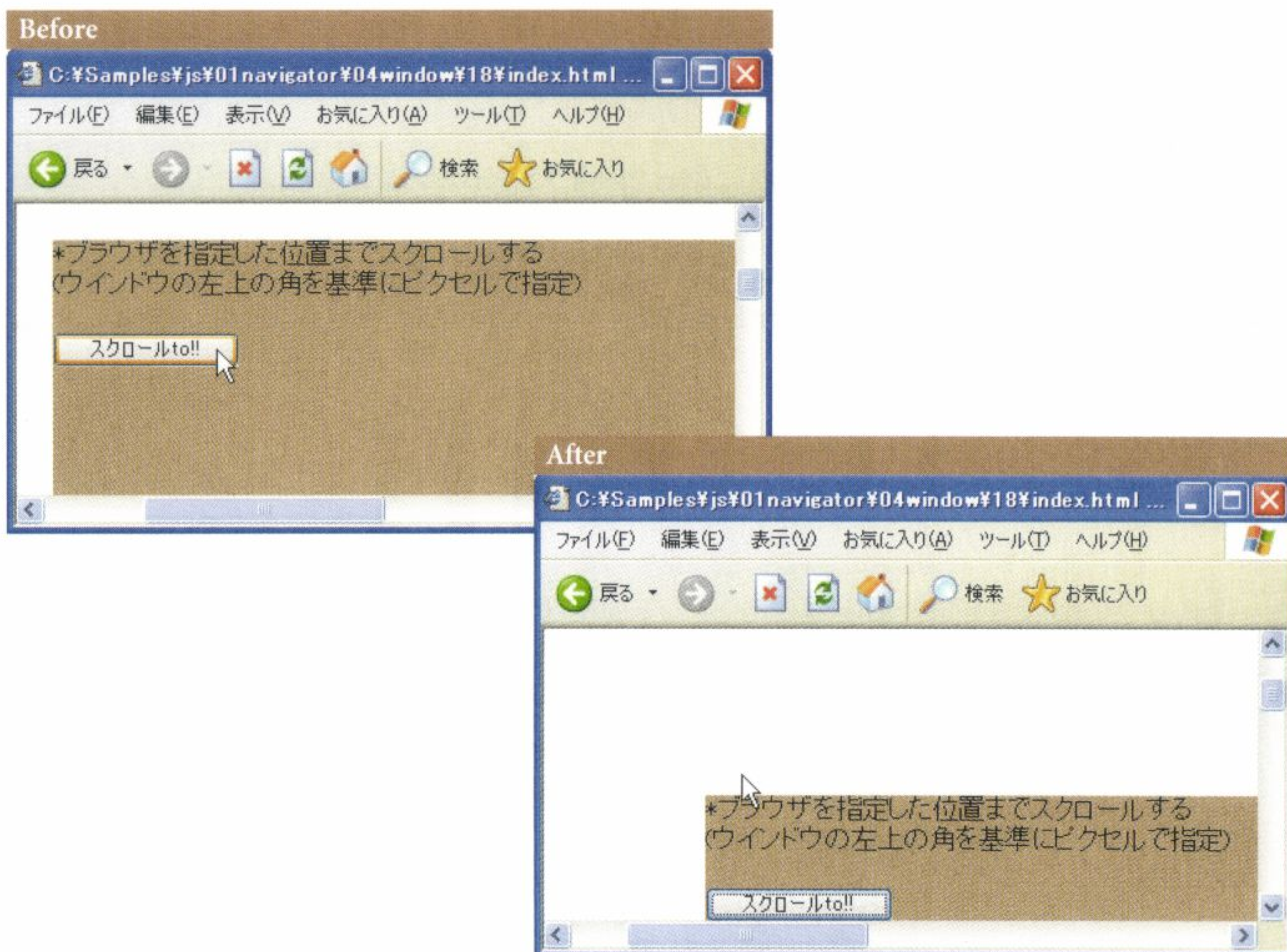
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body onLoad="SCLL2()">
<nobr>
<font size=6>                                        フレーム内の文字をスクロ
ールします...nobrの指定とsettimeoutがポイントです。for文を使うとスクロール中は何
もできなくなってしまいます。Windowsではスクロールバーを"no"にしているとうまくいかな
い時があります。</font>
</nobr>
</body>
</html>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# ブラウザを指定した位置までスクロールする

| | | |
|---|---|---|
| window.scrollTo(*x*,*y*) | | [メソッド] |
| x | | ウィンドウ左上角から垂直方向(ピクセル) |
| y | | ウィンドウ左上角から水平方向(ピクセル) |




「scrollTo()」メソッドは、JavaScript1.1の「scroll()」メソッドを拡張したもので、ピクセル単位で指定した位置へ、ウィンドウの表示領域を移動させます。
「scroll()」メソッドも以前までと互換を保つために使用可能になっています。
サンプルでは、[スクロールto!!]ボタンをクリックすると、ウィンドウの表示領域の左上角から、上へ100ピクセルで左へ100ピクセルの位置に移動し、[戻す!!]ボタンを押すと下の位置へ戻ります。次項の「scrollBy()」メソッドと違い、1度表示領域が移動すると、それ以降はボタンをクリックしても表示位置は変わりません。
Macintosh版のNetscape Navigator 4.X以外のブラウザの場合は、表示領域はスクロールバーが出ている範囲でしか移動しないので、サンプルでは1000×1000ピクセルの空領域を設定することによって、スクロールバーが出るようにしています。
JavaScript1.2で追加されたメソッドです。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function SRto1(){ window.scrollTo(100,100) }
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

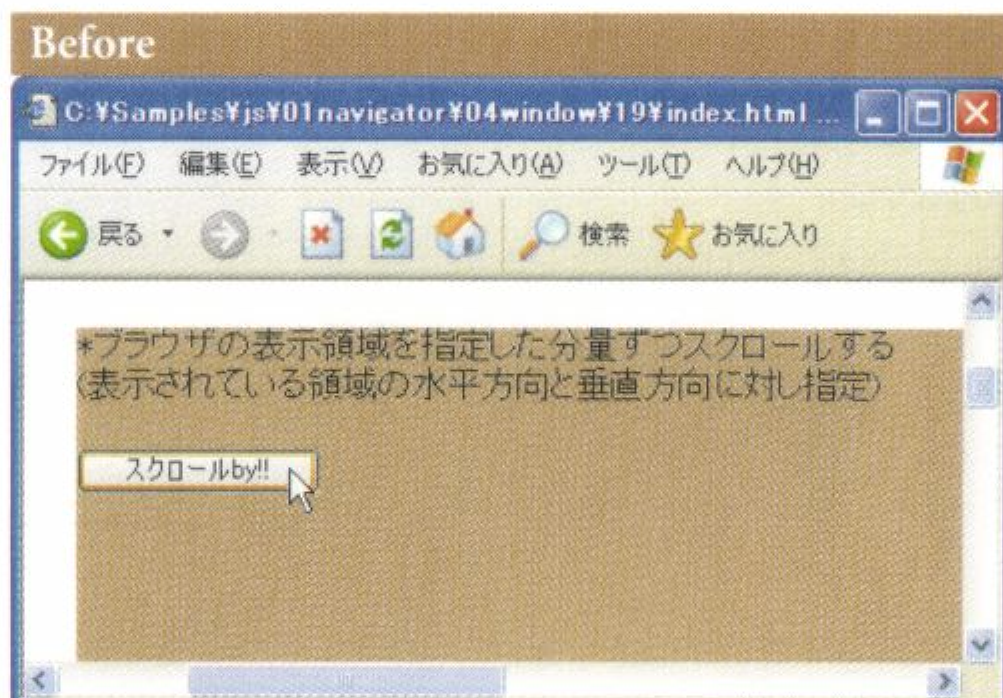
</head>
<body>
<div id="STY1" style="position:absolute; left:200px; top:200px; 
width:1000px; height:1000px; background: Tan">
*ブラウザを指定した位置までスクロールする<br>
（ウインドウの左上の角を基準にピクセルで指定）
<form>
    <input type="button" value=" スクロールto!! " onClick="SRto1()">
</form>
</div>
</body>
</html>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

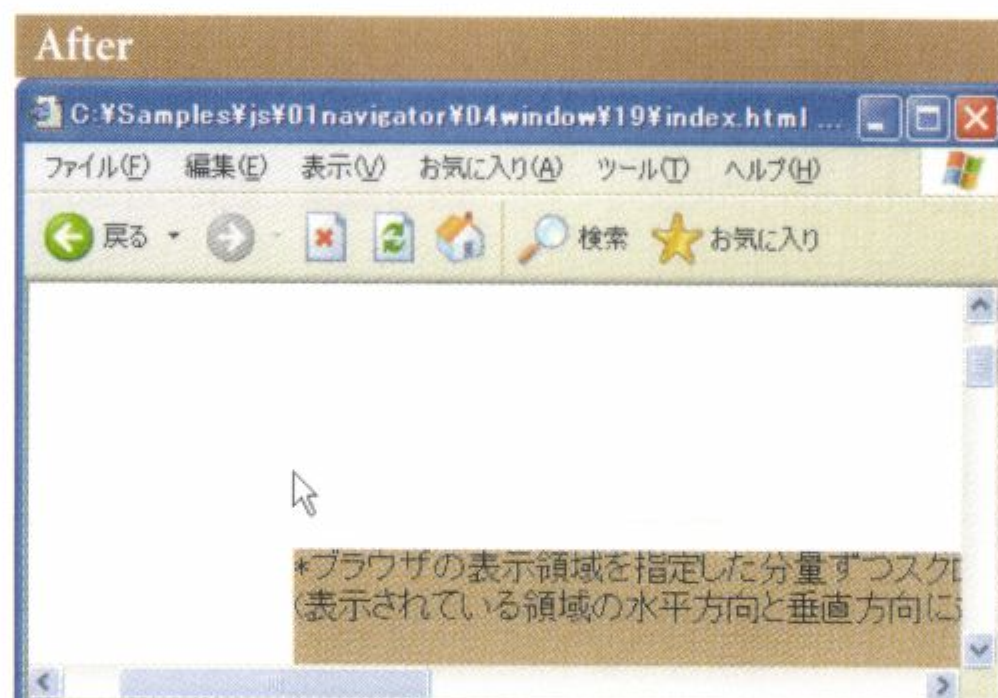
# ブラウザの表示領域を指定した分量ずつスクロールする

| | |
|---|---|
| **window.scrollBy(*x*,*y*)** | [メソッド] |
| x | 水平方向(ピクセル) |
| y | 垂直方向(ピクセル) |

Before




After




「scrollBy()」メソッドは、ウィンドウの表示領域をピクセル単位で移動します。
サンプルでは、ボタンをクリックするごとに、表示領域が右と下へ100ピクセルずつ移動します。
Macintosh版のNetscape Navigator 4.X以外のブラウザの場合、表示領域はスクロールバーが出ている範囲でしか移動しないので、サンプルでは1000×1000ピクセルの空領域を左端と上から200ピクセルの位置に配置してスクロールバーが出るようにし、そこにボタンを設置しています。なお、Macintosh版のNetscape Navigatorはスクロールバーが出ていない場合でもスクロール可能です。
JavaScript1.2で追加されたメソッドです。

## Sample

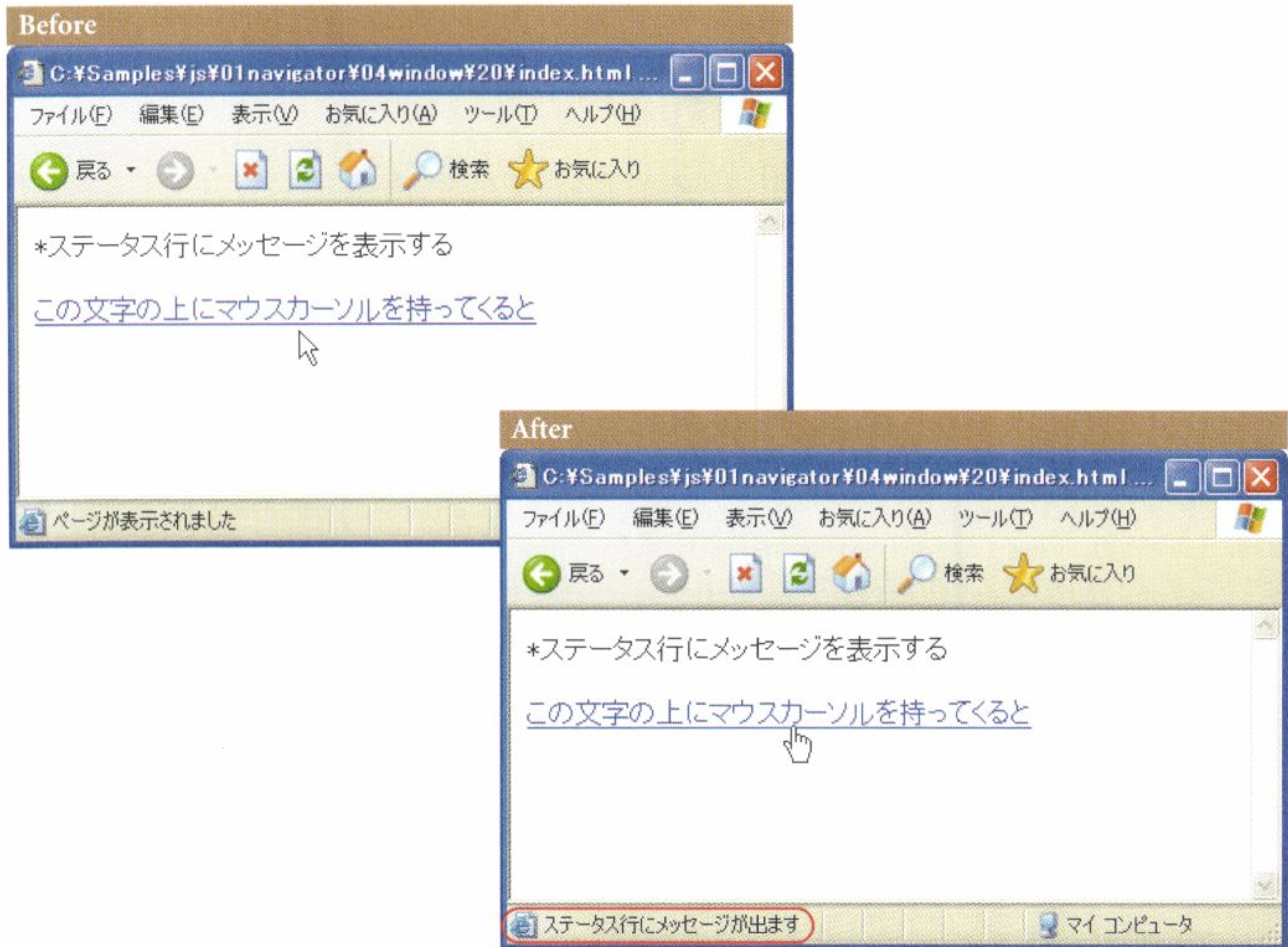
```
<script type="text/javascript">
<!--
function SRby(){ window.scrollBy(-100,-100) }
//-->
</script>
  ～中略～
<div id="STY1" style="position:absolute; left:200px; top:200px; 
width:1000px; height:1000px; background: Tan">
*ブラウザを指定した分量ずつスクロールする<br>
(表示されている領域の水平方向と垂直方向に対し指定)
<form>
    <input type="button" value=" スクロールby!! " onClick="SRby()">
</form>
</div>
```

ウィンドウを操作する

# ステータス行にメッセージを表示する

**window.status** [プロパティ]
**onMouseOver="*スクリプト | 関数*"** [イベントハンドラ]





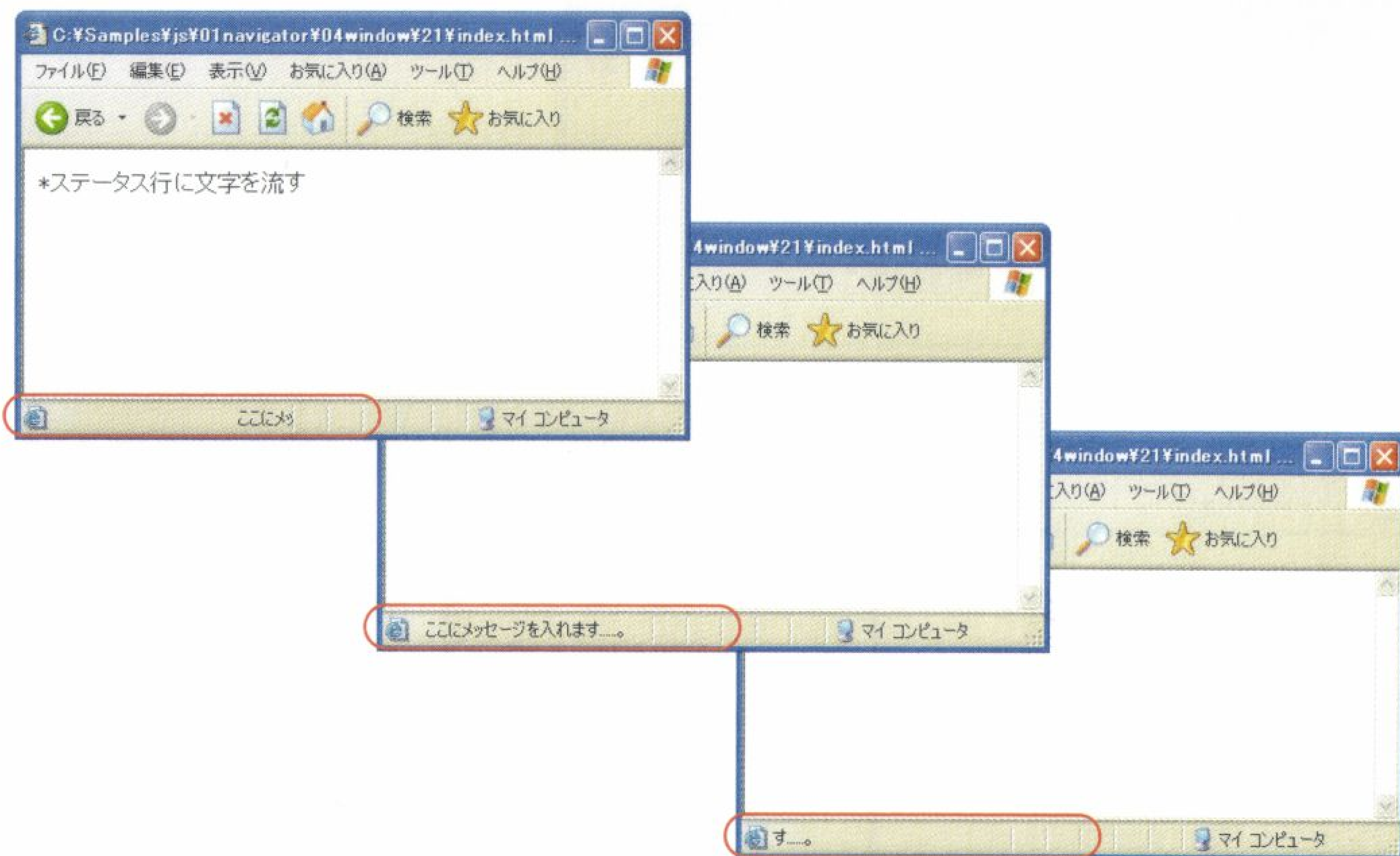

サンプルでは、リンク上にマウスポインタが乗った時にイベントハンドラ「onMouseOver」内のスクリプトが評価され、「window.status」によってステータス行に文字を表示させています。
リンクの説明を表示するなどの利用方法があります。
このスクリプトは、Netscape 6.2では動作しませんが、それ以前のNetscape 6.Xでは正常に動きます。

## Sample

```
<a href="#" onMouseOver=" window.status='ステータス行にメッセージが出ます'
;return true">
この文字の上にマウスカーソルを持ってくると
</a>
```

# ステータス行に文字を流す

**window.status** [プロパティ]
**setTimeout(*処理,時間設定*)** [メソッド]
**clearTimeout()** [メソッド]




サンプルでは、まずページが読み込まれた時に、イベントハンドラ「onLoad」が関数「Mess()」を発生します。
次に「Mess(){...}」内では、TCの値に1を加え、「window.status」でステータス行に文字列を書き出した後、stringオブジェクトの「substring()」メソッドで文字列を2字ずらし、「setTimeout()」メソッドで再び関数「Mess()」を呼び出しています。そして、この処理を「(TC<1000)」が真になるまで繰り返すことによって、ステータス行に文字が流れるような効果を出しています。
「Mess()」の処理が終った時は、ステータス行にスペースを表示することによって、ステータス行の文字をクリアしています。
文字列の最後と先頭がくっついたり、いきなり文字が現われたりしないように、文字列の始めにもスペースを入れています。
スクロールされている長さの調整は、「(TC < 1000)」内の数値を変更することによって行います。
スクロールの速度は、「setTimeout("Mess()",300)」内の数値を変えることによってミリ秒単位で調整します。
このスクリプトは、Mac OS 9.X上のNetscape 6.Xなど、一部の環境では動作しない場合があります。

**Sample**

```
<script type="text/javascript">
<!--
var TC = 0 ;
var Sm1 = "                                          ";
var Sm2 = "                                          ";
var Sm3 = "                                          ";
var Sm4 = "ここにメッセージを入れます......。";
var Smess = Sm1+Sm2+Sm3+Sm4;
var timeID=setTimeout("",1) ;
function Mess() {
    if (TC < 1000) { //ここの数値を変えることによってスクロールする時間が変わります
        TC++ ;
            window.status = Smess;
            Smess = Smess.substring(2,Smess.length) + Smess.substr
ing(0,2);
            clearTimeout(timeID);
            timeID = setTimeout("Mess()",300);
    }
    else { window.status = " " }
}
//-->
</script>

  ～中略～
</head>
<body onLoad="Mess()">
*ステータス行に文字を流す
</body>
```

substring()：「stringオブジェクト」の「文字列の途中の文字を抜き出す」(P.556)
onLoad：リファレンス「イベントハンドラ」の「onLoad」(P.620)

**注意**

## 「setTimeout()」メソッドの処理を繰り返した時の注意

Netscape Navigator 2.0で「setTimeout()」メソッドの処理を繰り返したJavaScriptを実行した時、しばらくすると「メモリーエラー」が発生し、ブラウザの動作が不安定になってしまいます。

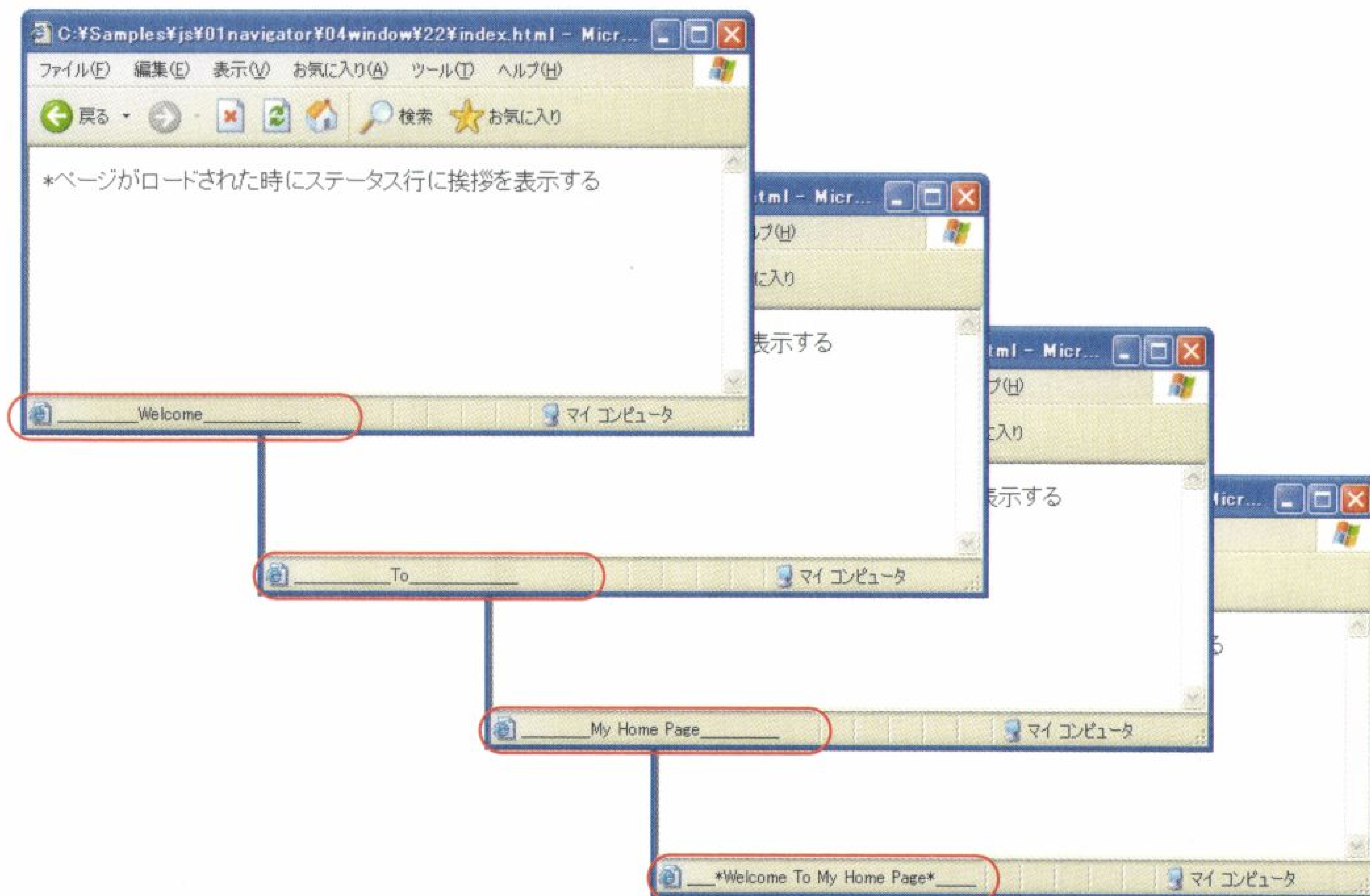
本書では、このエラーを回避するため、一定時間で処理を終了させるという対策をとっています。

その他にも、エラー回避のために「clearTimeout()」メソッドを使って、1回1回「setTimeout()」の処理を明示的に終了させるなどの対策があります。

このエラーは、Netscape Navigator 3.0以降では発生しません。

# ページがロードされた時にステータス行に挨拶を表示する

**window.status** [プロパティ]
**setTimeout(*処理,時間設定*)** [メソッド]




サンプルでは、ステータス行に表示する文字列の配列を作り、ページが読み込まれた時に、配列の要素を順番にステータス行に書き出しています。
表示の処理が終った時には、ステータス行にスペースを表示することによって、ステータス行の文字をクリアしています。
表示のタイミングは、「setTimeout("EVENT3()", 500)」内の数値を変えることによってミリ秒単位で調整します。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
var TC = 0 ;
var j = 0
function MakeArray(n) {
```

```
    this.length = n ;
    for (var i = 0; i <= n; i++) {
        this[i] = 0;
    }
    return this ;
}
msg = new MakeArray(16) ;
msg[0] = " " ;
msg[1] = " " ;
msg[2] = "______________Welcome_________________" ;
msg[3] = "______________Welcome_________________" ;
msg[4] = "______________________________________" ;
msg[5] = "_________________To___________________" ;
msg[6] = "_________________To___________________" ;
msg[7] = "______________________________________" ;
msg[8] = "____________My Home Page______________" ;
msg[9] = "____________My Home Page______________" ;
msg[10] = "                                     " ;
msg[11] = "                                     " ;
msg[12] = "                                     " ;
msg[13] = "_____*Welcome To My Home Page*_______" ;
msg[14] = "_____*Welcome To My Home Page*_______" ;
msg[15] = "_____*Welcome To My Home Page*_______" ;
function EVENT3() {
    if (TC < 16) {
        TC++ ;
            if (j <= msg.length) {
            window.status = msg[j] ;
            j++ ;
            if (j == msg.length) {
                j = 0 ;
            }
            setTimeout("EVENT3()", 500);
        }
    } else {
        window.status = " " ;
    }
}
//-->
</script>
</head>
<body onLoad="EVENT3()">
＊ページがロードされた時にステータス行に挨拶を表示する
</body>
</html>
```

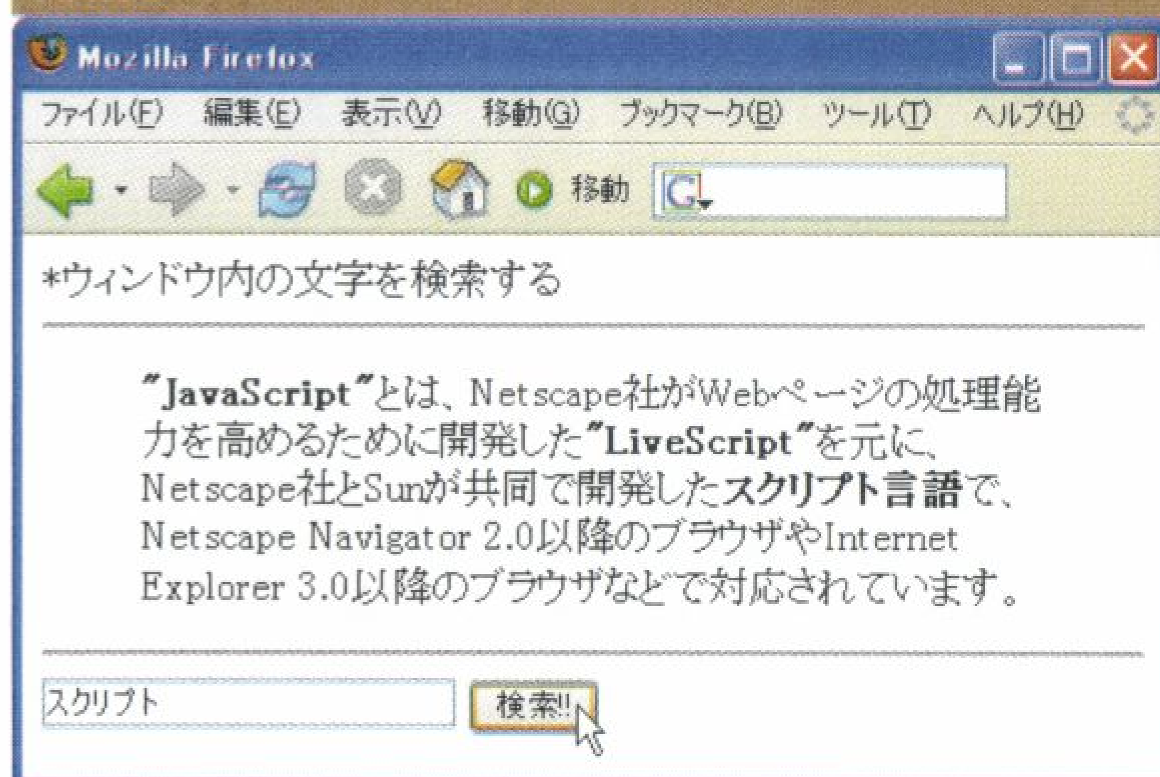

length：「複数のオブジェクトで使用できるプロパティ・メソッド」の「オブジェクト（配列）の数を取得する」（P.580）

# ウィンドウ内の文字を検索する

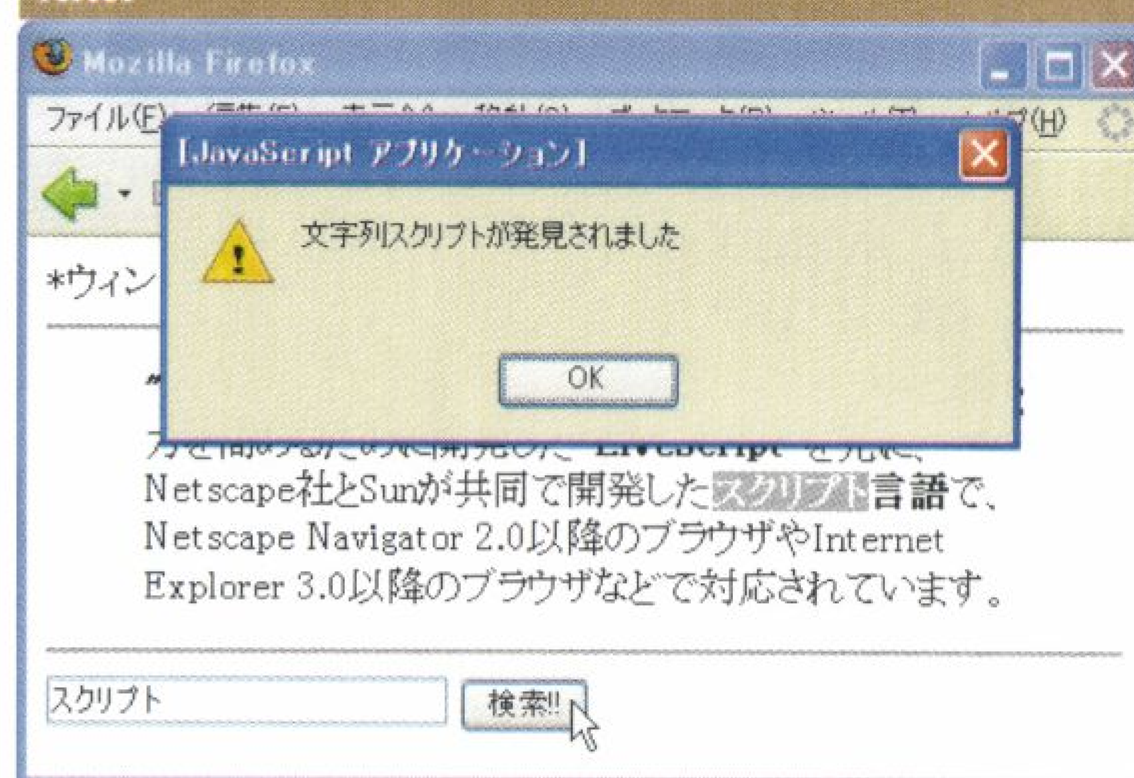
**window.find(文字列,[*true* | *false*])** [メソッド]



Before




After




「find()」メソッドは、ウィンドウ内の文字列を検索し、指定された文字列が発見された時に、真(true)または偽(false)を返すように設定することができます。
サンプルでは、フォームに入力された文字列がウィンドウ上に含まれているかどうかを検索し、結果を警告用のダイアログボックスに表示しています。
JavaScript1.2で追加されたメソッドです。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
```

```
<script type="text/javascript">
<!--
function FIN(i) {
        if ( window.find( i,true) ) { alert ("文字列" + i + "が発見
されました") }
       else { alert ("文字列" + i + "は発見されませんでした") }
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*ウィンドウ内の文字を検索する
<hr>
<blockquote>
<b>"JavaScript"</b>とは、Netscape社がWebページの処理能力を高めるために開発した
<b>"LiveScript"</b>を元に、Netscape社とSunが共同で開発した<b>スクリプト言語
</b>で、Netscape Navigator 2.0以降のブラウザやInternet Explorer 3.0以降
のブラウザなどで対応されています。
</blockquote>
<hr>
<form>
    <input type="text" name="fin1" value="" size=30>
    <input type="button" name= "fin2" value=" 検索!! " onClick="FIN
(fin1.value)">
</form>
</body>
</html>
```

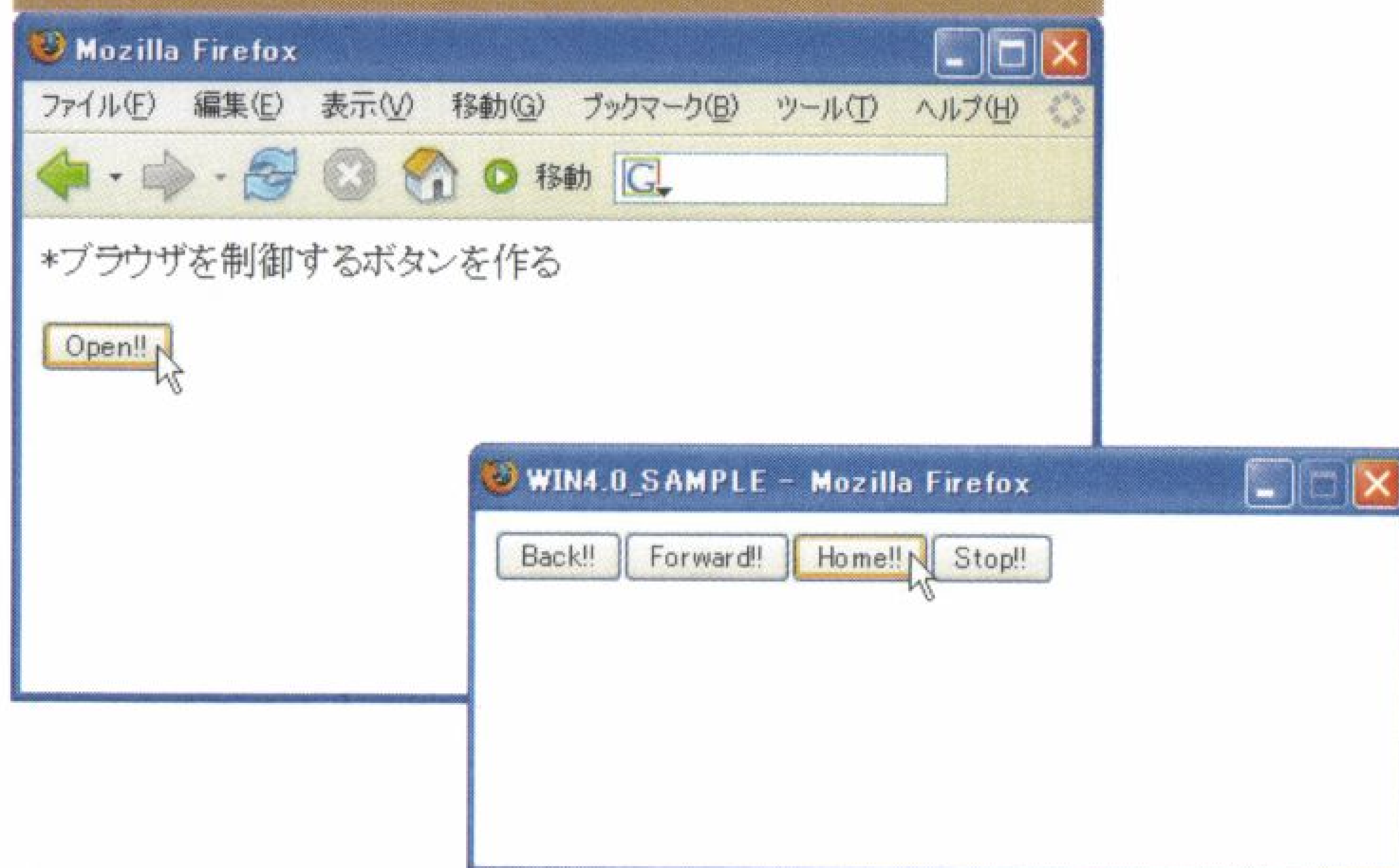
Firefox
N7.X
N4.06
N4.X
ウィンドウを操作する

# ブラウザを制御するボタンを作る

opener.back() ←[戻る]ボタン [メソッド]
opener.forward() ←[進む]ボタン [メソッド]
opener.home() ←[ホーム]ボタン [メソッド]
opener.stop() ←[中止]ボタン [メソッド]



Before




After






「back()」メソッドは、ブラウザの[戻る(Back)]ボタンで参照されるのと同じひとつ前のウィンドウのURLを返します。同様に、「forward()」メソッドはブラウザの[進む(Forward)]ボタンで参照されるのと同じひとつ先のウィンドウのURLを、「home()」メソッドはブラウザの[ホーム(Home)]ボタンで参照されるのと同じURLを、それぞれ返します。「stop()」メソッドは、ブラウザの[中止(Stop)]ボタンを押した時と同じ状態を返します。

サンプルでは、ブラウザのメニューバーのボタンと同じ働きを持ったボタンがあるサブウィンドウを開き、そこから元のウィンドウを操作できるようにしています。

JavaScript1.2で追加されたメソッドです。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function NewWin(){
        var NW;
        NW=window.open("","NewWin1","toolbar=no,location=no,direc
tories=no,Width=400,Height=150");
        NW.document.open();
        NW.document.write("<html><head><title>WIN4.0_SAMPLE</titl
e>");
        NW.document.write("</head>");
        NW.document.write("<body>");
        NW.document.write("<form>");
        NW.document.write("<input type='button' value=' Back!! ' on
Click='opener.back()'>");
        NW.document.write("<input type='button' value=' Forward!! '
onClick='opener.forward()'>");
        NW.document.write("<input type='button' value=' Home!! ' on
Click='opener.home()'>");
        NW.document.write("<input type='button' value=' Stop!! ' on
Click='opener.stop()'>");
        NW.document.write("</form>");
        NW.document.write("</body>");
        NW.document.write("</html>");
        NW.document.close();
}
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*ブラウザを制御するボタンを作る
<p>
<form>
    <input type="button" value=" Open!! " onClick="NewWin()">
</form>
</p>
</body>
</html>
```

# 各種バーを制御する

window.locationbar.visible = [true|false]|[1|0]　[プロパティ]
window.menubar.visible = [true|false]|[1|0]　[プロパティ]
window.personalber.visible = [true|false]|[1|0]　[プロパティ]
window.scrollbars.visible = [true|false]|[1|0]　[プロパティ]
window.statusbar.visible = [true|false]|[1|0]　[プロパティ]
window.toolbar.visible = [true|false]|[1|0]　[プロパティ]



JavaScript1.2対応のNetscape Navigator 4.0では、メニューバーやステータスバーなどの各種バーをプロパティとして持っています。
各種バーは、「window.locationbar.visible=true」や「window.locationbar.visible=1」というように、「.visible=[true | 1 ]」と指定されている時は表示状態になります。「window.locationbar.visible=false」や「window.locationbar.visible=0」というように、「.visible=[false | 0 ]」と指定されている時は非表示状態になります。これらのプロパティの値は、後から変更可能です。
これらのプロパティは、「Signed Script」内で設定する必要があります。
JavaScript1.2で追加されたプロパティです。

## コラム

### JavaScriptとポップアップ ブロック

Windows XP Service Pack 2では、セキュリティ関連の多くのアップデートがありました。その中でも、Internet Explorer 6 Service Pack 2に新たに搭載されたポップアップ ブロックの機能は、JavaScriptのを使う上で注意が必要なアップデートです。

ポップアップ ブロックとは、ポップアップウィンドウと呼ばれる、ブラウジング中に自動的に開かれる新しいウィンドウを、ユーザの許可なしには開かないようにする機能です。Windows XP Service Pack 2をあてたInternet Explorer 6では、標準でこのポップアップ ブロックが機能する状態になっています。

Internet Explorer 6の場合は、ポップアップ ブロックに引っ掛かると、まずはページ上部の情報バーやステータスバーにポップアップ ブロックが機能したことを表示します。そして、その部分をクリックすると、ポップアップウィンドウの表示を許可するかどうかを確認するメニューが表われるので、そこでポップアップを開くかどうかを選択するのです。また、［ツール］メニューから、ポップアップ ブロックを無効にしたり、特定のサイトのポップアップウィンドウは許可するように設定することもできるようになっています。

このポップアップ ブロックの機能は、Netscape 7やOpera、Firefoxなどのブラウザでは、以前から搭載されていました。これらのブラウザは、設定の項目にポップアップ ブロックを有効にするかどうかの設定があります。また、Internet Explorer 6と同じように、特定のウィンドウのポップアップを許可するように設定することもできます。

**コラム**

## JavaScriptとセキュリティ

JavaScriptは、セキュリティを確保するため、本来はブラウザ以外のユーザーのローカル資源に勝手にアクセスすることを許されていません。実際、JavaScriptが直接アクセスを許されている唯一のローカル資源は、cookieファイルのみです。このような、「影響を与える範囲をブラウザ内に限定し、それを越える範囲での動作を許さない」という思想は、JAVAの思想を受け継いでいる部分でもあります。

このためJavaScriptでは、ウイルスを埋め込むような動作や、問題を起こすプログラムを実行する、ブラウザ上以外のPC上のデータが漏れる、といったセキュリティ上の問題は、バグは別としても、基本的に起こりにくい作りになっているといえます。

しかし、これがInternet ExplorerとInternet Explorerが搭載しているJScriptとなると、話は少し違ってきます。

Windows版のInternet Explorerは、OS自身やOSが持っている機能と深い部分で結び付いています。これは、本来はOSとブラウザの結び付きを強めることによって、ユーザーの利便性を高めることが目的であったはずです。ところが、そのために、JScriptとActiveXなどの機能が連係しあうことによって、問題を起こすようなプログラムを埋め込んだり、実行したり、PC上のデータを漏らすような動作をしてしまう場合があります。つまり、この場合の利便性は、現時点では、セキュリティ上のぜい弱性につながってしまっているようなのです。

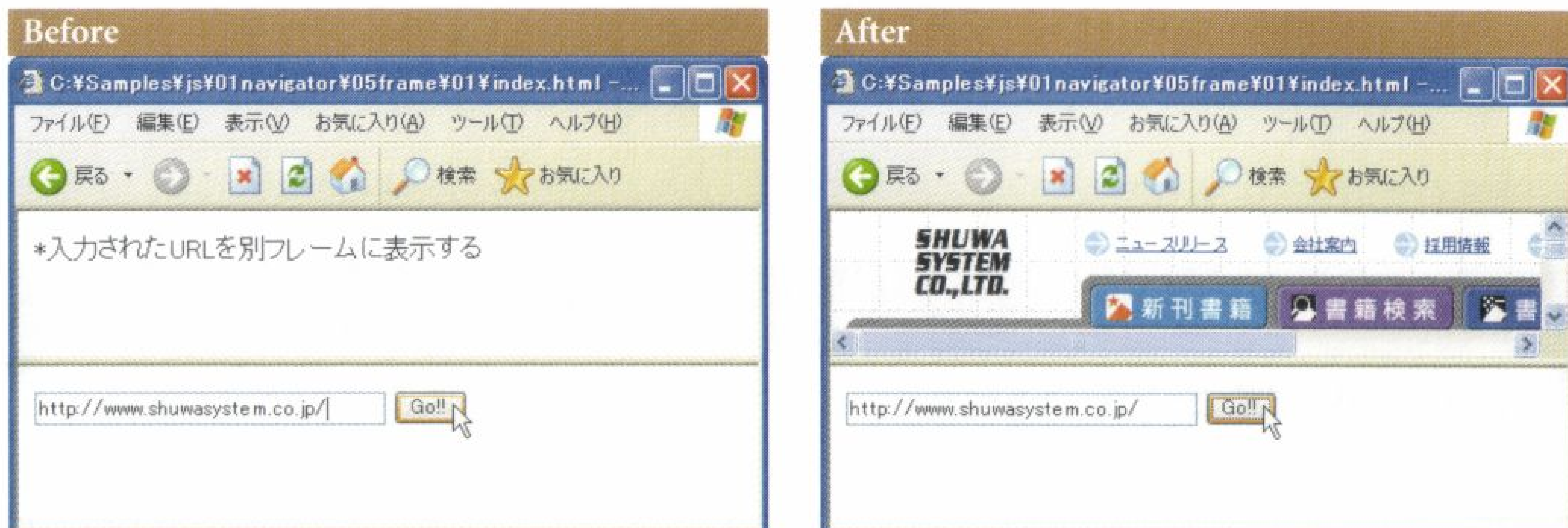
もちろん、それなりの対策はとられてはいますが、このようなOSとアプリケーションが深く統合された環境では、問題を完全に取り除くのは難しいのではないかと思われます。また悲しいことに、実際にInternet Explorerのこのような特徴を利用したウイルスも、多数生まれてきています。

また、JScriptにある「ユーザーの利便性を高める」という目的で独自に拡張された機能の中には、クリップボードの内容を貼り付ける「clipboardData」といったものがあります。これは、悪意を持って使われた場合、データ漏えいの観点から見れば、かなり危険な機能です。

このことからも、もしもWindowsとInternet Explorerの環境でインターネットを利用しているのであれば、[ツール]メニューにある[インターネットオプション]の[セキュリティ]の設定内容を良く理解し、自分にあったセキュリティレベルの設定を行う必要があるといえるでしょう。

# 入力されたURLを別フレームに表示する

## parent.フレーム名.プロパティ




JavaScriptを記述しているフレーム以外のフレームにJavaScriptの値を引き渡す時は、「parent.値を渡すフレーム名.値」と指定します。
サンプルでは、フォーム内に入力されたURLの値を「parent.f1.location」として、フレームネーム「f1」の「ロケーションの値」に引き渡しています。
一見ややこしそうですが、慣れれば非常に簡単で利用価値の高い用法なので、ぜひ1度チャレンジしてみてください。
なお、「location」プロパティの使い方に関しては、「locationオブジェクト」の「入力されたURLへ進むフォームを作る」(P.409)を参照してください。
また、フレームを抜けてページを表示するには、「parent.top.location.href=URL」と「top」プロパティを指定します。具体的な使い方は、「Formオブジェクト」の「ラジオボタンをリンクに使う」(P.424)を参考にしてください。
実際に試す場合には、この他にも「f1.html」を用意してください。

## Sample

**【フレームウィンドウ】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" >
<html>
<head>
<title></title>
</head>
<frameset rows="*,100">
    <frame src="f1.html" name="f1">
    <frame src="f2.html" name="f2">
</frameset>
<noframes>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</noframes>
</html>
```

**【f2.html】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function LC2(go){
    if (go.url2.value != "") { parent.f1.location.href=go.url2.val
ue }
    else { alert("URLを入力してください") }
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
<form name="URL2">
<input type="text" name="url2" value="http://" size=40 >
<input type="button" name= "CF2" value=" Go!! "
onClick="LC2(this.form)">
</form>
</body>
</html>
```


<frameset>：「フレーム」の「フレームの全体構造を指定する」(P.135)
<noframes>：「フレーム」の「フレームが表示されない環境用の内容を入れる」(P.143)
location：「locationオブジェクト」の「入力されたURLへ進むフォームを作る」(P.409)
top：「Formオブジェクト」の「ラジオボタンをリンクに使う」(P.424)

**重要**

## JavaScriptで操作できるフレームの範囲

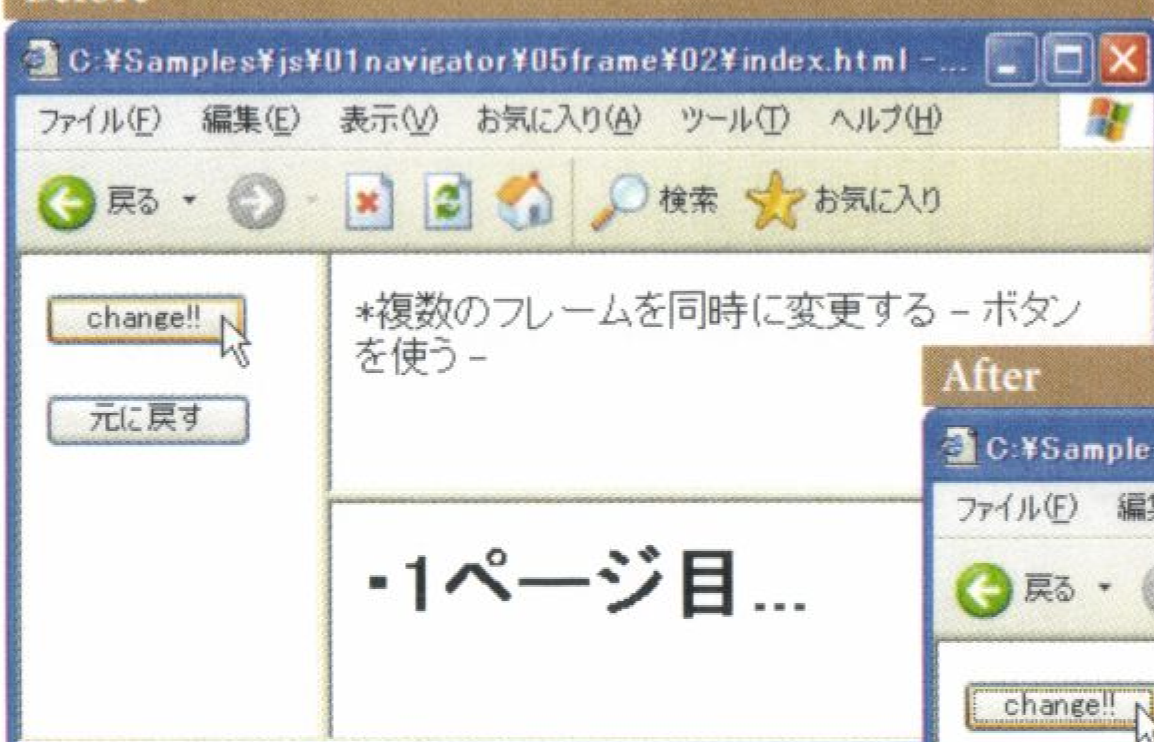
JavaScriptでは、セキュリティー上の配慮から、親フレームと子フレーム、あるいは複数のフレームに表示したHTMLファイルが別ドメインの場合は、子フレーム間で値を取得したり、操作することはできません。

JavaScript

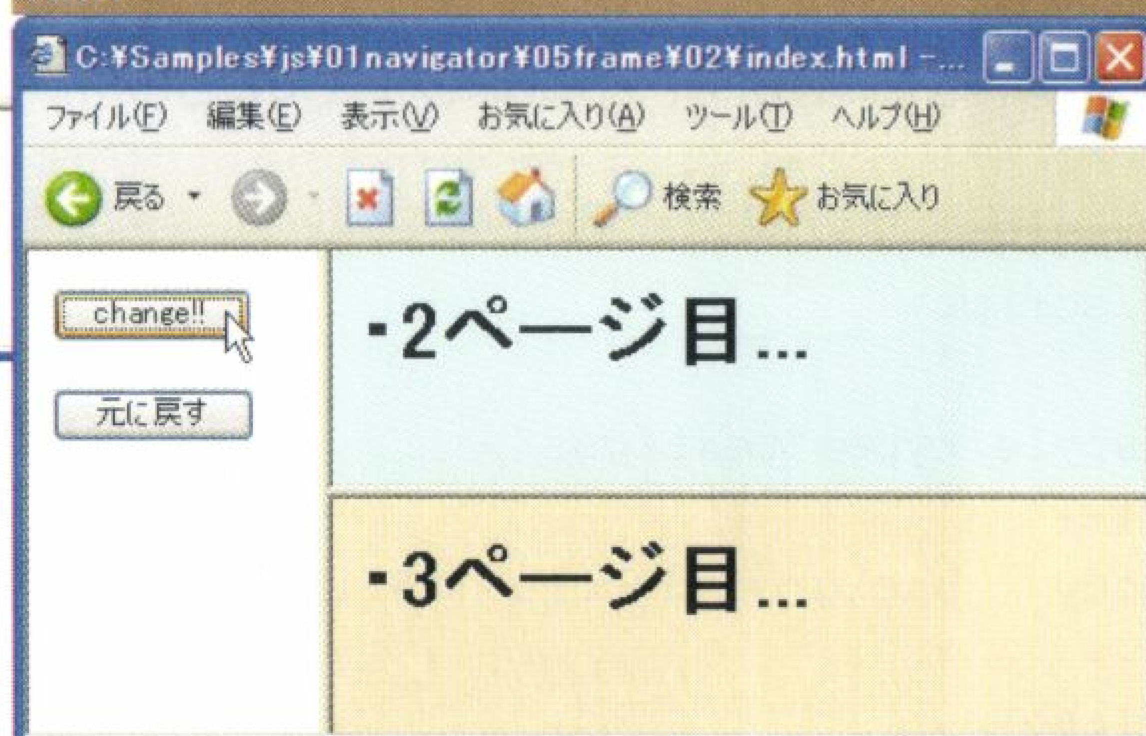
# 複数のフレームを同時に変更する - ボタンを使う -

**parent.フレーム名.location.href=*URL***

Before




After




サンプルでは、ボタンがクリックされた時に、ふたつのURLの値を持った関数を発生し、その値を1度に関数の処理へ引き渡すことにより、複数のフレームに新しいページを読み込んでいます。このように、「parent.フレーム名.location.href=URL」の処理を複数設定するだけで、1回の処理で複数のフォームを変更することが可能です。
実際に試す場合には、この他にも「page1.html」〜「page3.html」の3つのHTMLファイルを用意してください。

## Sample

**【フレームウィンドウ】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" >
<html><head>
<title></title>
</head>
<frameset cols="120,*">
    <frame src="f1.html" name="f1">
  <frameset rows="50%,50%">
    <frame src="f2.html" name="f2">
    <frame src="page1.html" name="f3">
   </frameset>
</frameset>
<noframes>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</noframes>
</html>
```

【f1.html】

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function CH1(P1,P2){
    parent.f2.location.href=P1;
    parent.f3.location.href=P2;
}
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
<p>
<form>
<input type="button" value=" change!! " onClick="CH1('page2.html',
'page3.html')">
</form>
</p>
<p>
<form>
<input type="button" value=" 元に戻す " onClick="CH1('f2.html','page
1.html')">
</form>
</p>
</body></html>
```

【f2.html】

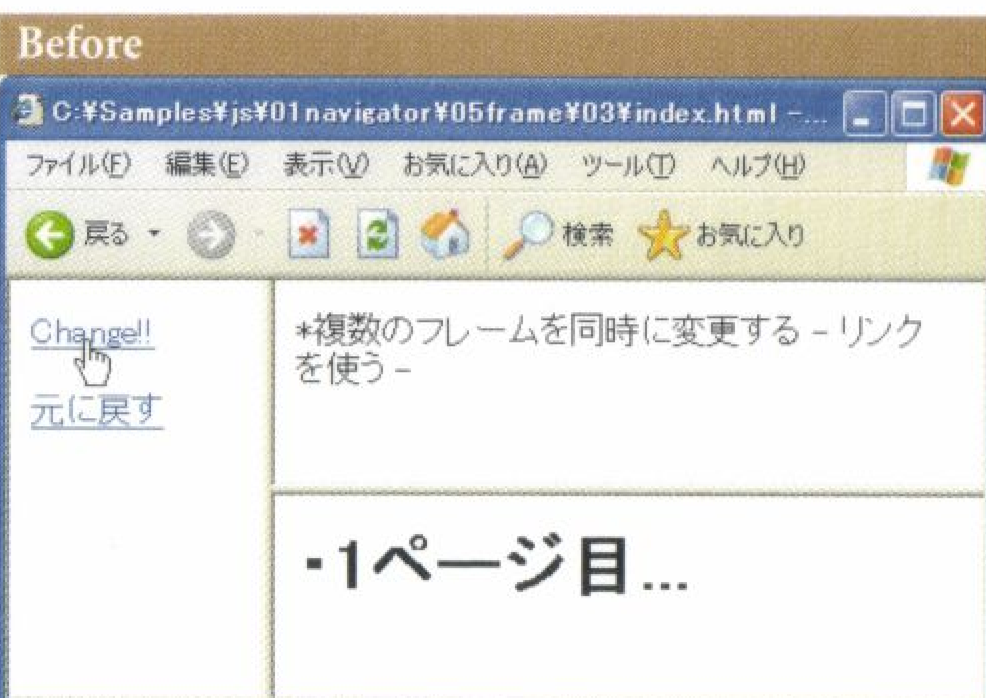
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
  ～中略～
</head>
<body>
*複数のフレームを同時に変更する - ボタンを使う -
</body></html>
```

 <frameset>：「フレーム」の「フレームの全体構造を指定する」(P.135)
<noframes>：「フレーム」の「フレームが表示されない環境用の内容を入れる」(P.143)
<input type="button">：「formオブジェクト」の「ボタンをリンクに使う」(P.426)

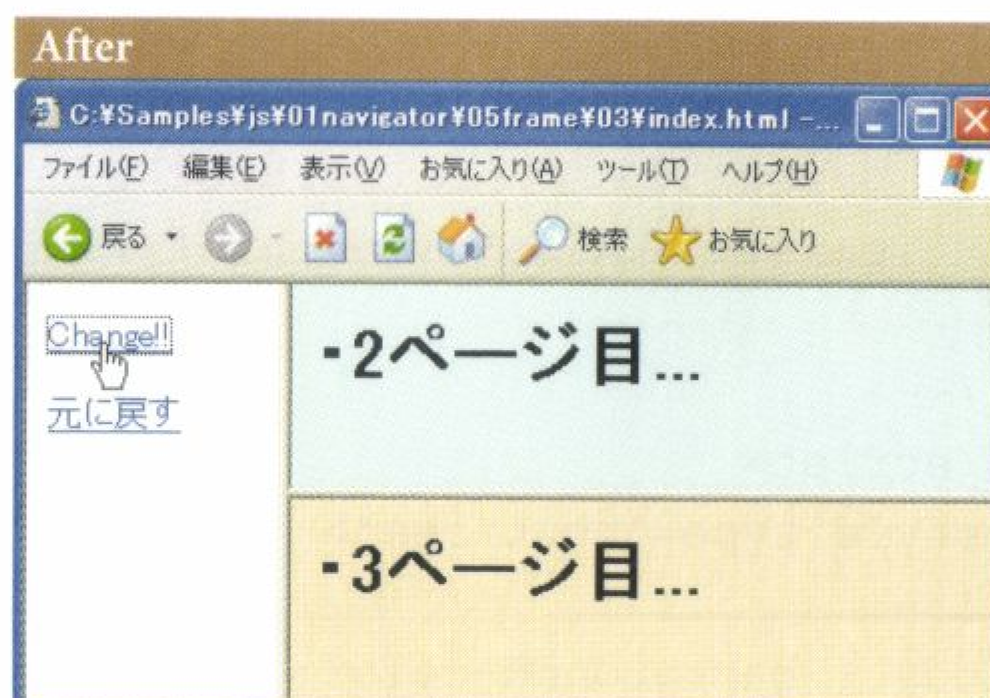
# 複数のフレームを同時に変更する - リンクを使う -

**window.open(*URL*,フレーム名)**
**href="JavaScript:*関数*"**

Before




After




サンプルの「window.open(URL,フレーム名)」の部分は、frameオブジェクトというよりは、windowオブジェクト的な用法です。
サンプルでは、リンクがクリックされた時にふたつのURLの値を持った関数を発生して、その値を1度に関数の処理へ引き渡すことにより、それぞれのフレームに新しいウィンドウを開いています。この時に、本来ウィンドウ名に当る部分は、フレーム名として取り扱われます。
実際に試す場合には、この他にも「f2.html」「page1.html」～「page3.html」の4つのHTMLファイルを用意してください。
なお、フレームを抜けてページを表示するには、「window.open(URL,"_top")」と、ウィンドウ名に「_top」を指定します。具体的な使い方は、「Formオブジェクト」の「ラジオボタンをリンクに使う」(P.424)を参考にしてください。
また、「href="JavaScript:関数"」の使い方は、「Link・Anchorオブジェクト」の「リンクをボタンのように使う - 1 -」(P.420)を参考にしてください。

## Sample

**【フレームウィンドウ】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" >
<html>
<head>
<title></title>
  ～中略～
</head>
<frameset cols="120,*">
    <frame src="f1.html" name="f1">
  <frameset rows="50%,50%">
    <frame src="f2.html" name="f2">
    <frame src="page1.html" name="f3">
   </frameset>
```

```
</frameset>
<noframes>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</noframes>
</html>
```

**【f1.html】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function CH1(P1,P2){
    window.open(P1,"f2");
    window.open(P2,"f3");
}
//-->
</script>
 ～中略～
</head>
<body>
<p>
<a href="JavaScript:CH1('page2.html','page3.html')">Change!!</a>
</p>
<p>
<a href="JavaScript:CH1('f2.html','page1.html')">元に戻す</a>
</p>
</body></html>
```


<frameset>：「フレーム」の「フレームの全体構造を指定する」(P.135)
<noframes>：「フレーム」の「フレームが表示されない環境用の内容を入れる」(P.143)
window.open()：「windowオブジェクト」の「新しいウィンドウを開く」(P.338)
href="JavaScript:関数"：「Link・Anchorオブジェクト」の「リンクをボタンのように使う - 1 -」(P.420)
top：「Formオブジェクト」の「ラジオボタンをリンクに使う」(P.424)

**注意**

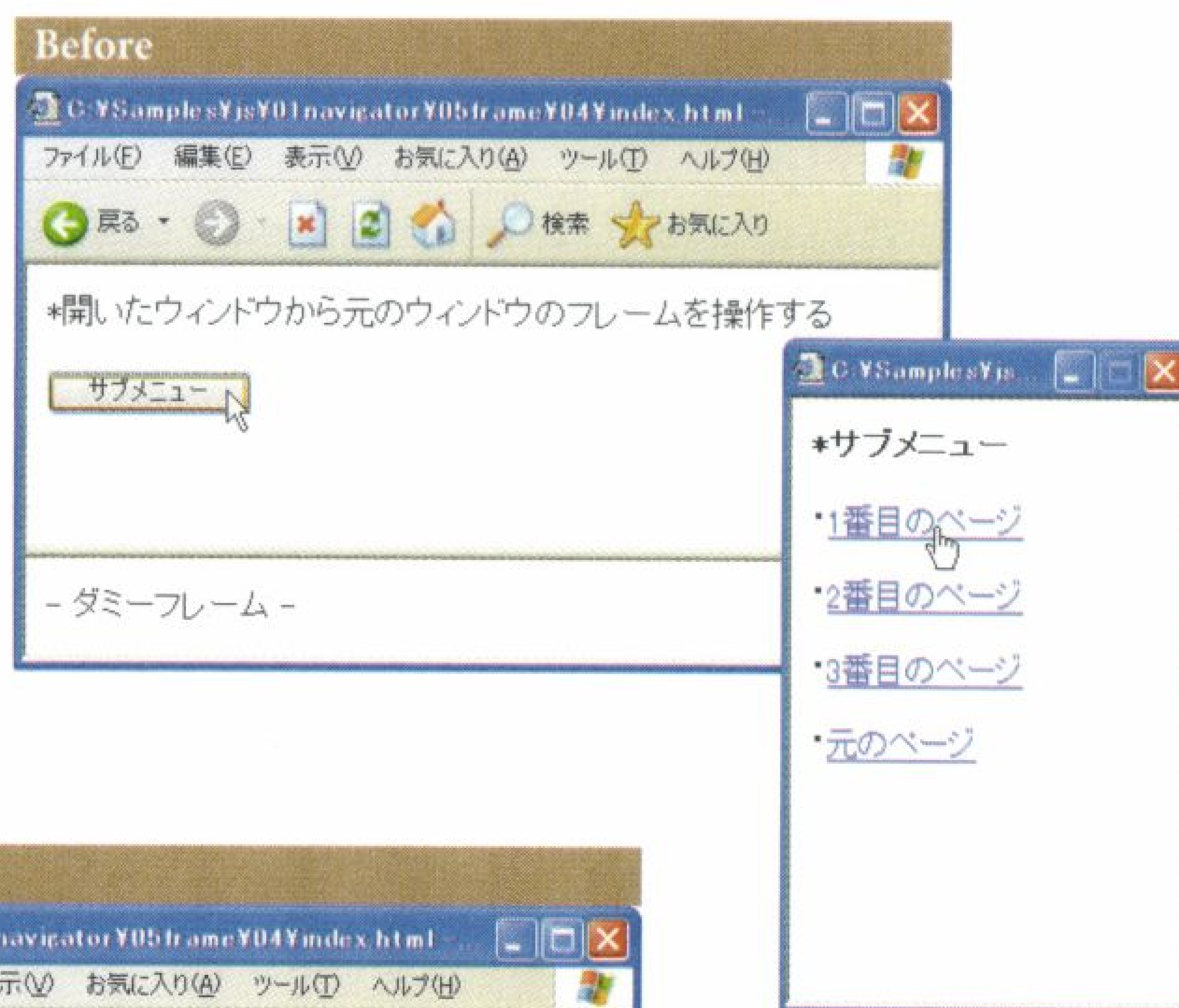
### フレーム内にページをロードする時の注意

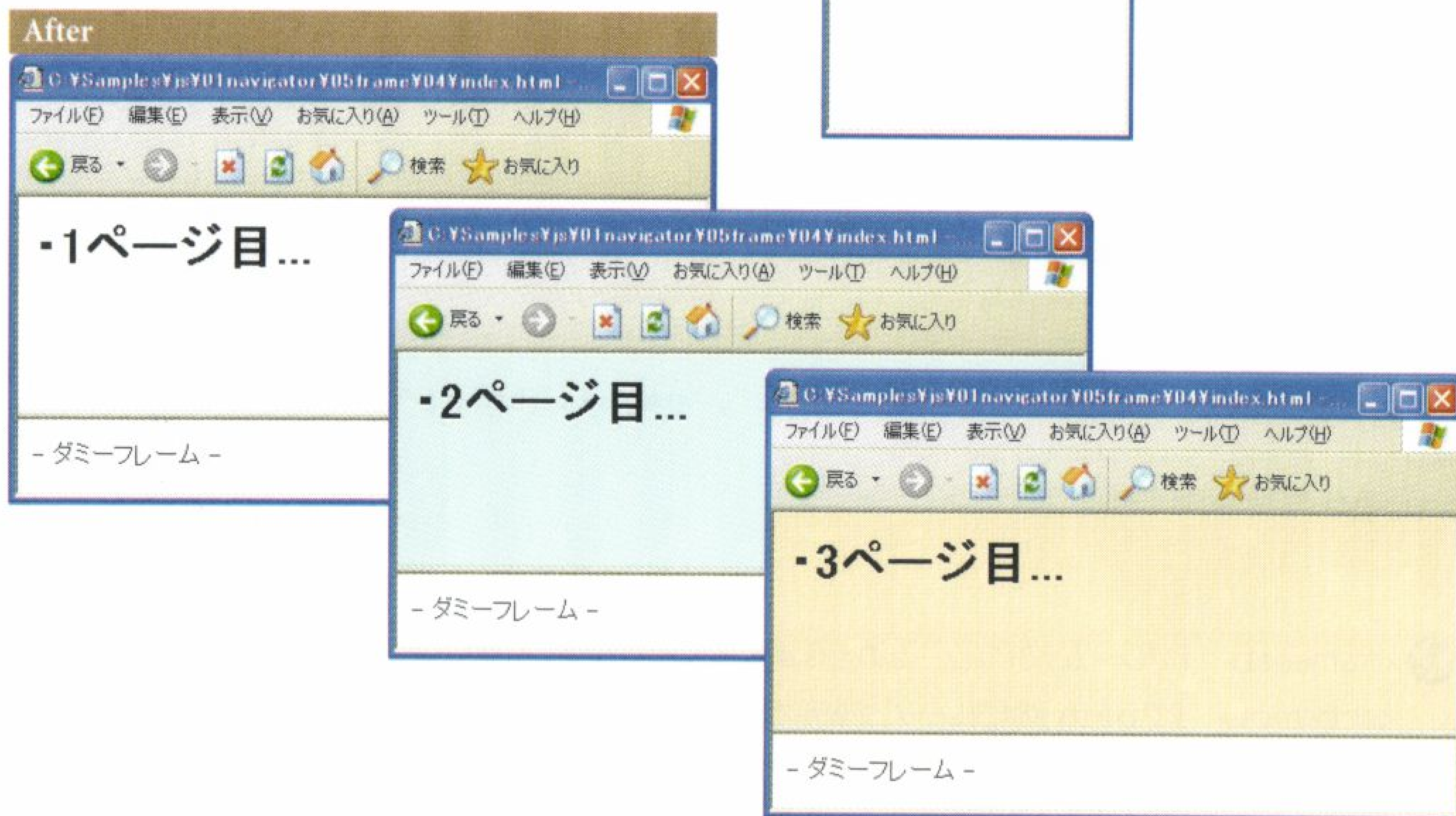
frameオブジェクトを使用して「parent.フレーム名.location.href=URL」としてフレームに新しいページをロードしようとした場合、Macintosh版のNetscape Navigator 2.0などの一部ブラウザでは、ふたつ目以降のURLが引けずにエラーになってしまうことがあります。

そのような場合は、上のサンプルを参照してください。

# 開いたウィンドウから元のウィンドウのフレームを操作する

**window.open()**
**target=** フレーム名




サンプルでは、ウィンドウ名とフレーム名が、同じように取り扱われることを利用しています。

フレーム内から新しいウィンドウを開き、そのウィンドウから「<a href="URL" target= フレーム名>」というように、ターゲットウィンドウの設定にフレーム名を指定することにより、指定したフレームにページをロードしています。

実際に試す場合には、この他にも「page2.html」と「page3.html」のふたつのHTMLファイルを用意してください。

「windowオブジェクト」の「開いたウィンドウから元のウィンドウを操作する」(P.343)はNetscape Navigator 3.0以降でしか使えませんが、このサンプルはNetscape Navigator 2.0でも正常に作動します。

## Sample

**【フレームウィンドウ】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" >
<html>
<head>
<title></title>
</head>
<frameset rows="*,50">
    <frame src="f1.html" name="f1">
    <frame src="f2.html" name="f2">
</frameset>
<noframes>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</noframes>
</html>
```

**【f1.html】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function wopen(){
    var WO1;
    WO1=window.open("menu1.html","MenuWindow",
                    "toolbar=no,location=no,directories=no,sta
tus=no,menubar=no,scrollbars=no,resizable=no,width=200,height=
300");
}
//-->
</script>
//-->
</script>
  ～中略～
</head>
<body>
*開いたウィンドウから元のウィンドウのフレームを操作する
<p>
<form name="MENU1">
<input type="button" name= "nenu1" value=" サブメニュー " onClick="wopen()">
</form>
</p>
</body>
</html>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

**【f2.html】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
  ～中略～
</head>
<body>
- ダミーフレーム -
</body></html>
```

**【menu1.html】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
  ～中略～
</head>
<body>
<b>*サブメニュー</b>
<p>
<p>・<a href="page1.html" target=f1>1番目のページ</a>
<p>・<a href="page2.html" target=f1>2番目のページ</a>
<p>・<a href="page3.html" target=f1>3番目のページ</a>
<p>・<a href="f1.html" target=f1>元のページ</a><p>
</p>
</body></html>
```

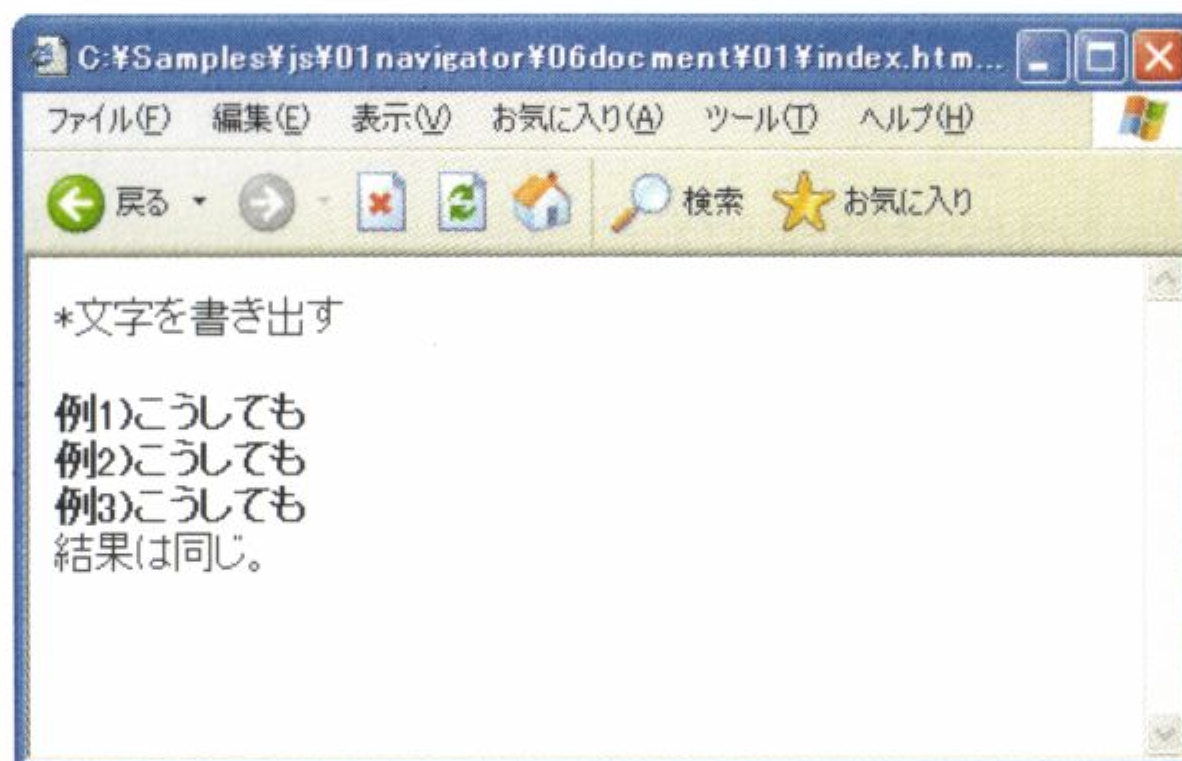
**【page1.html】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
  ～中略～
</head>
<body>
<h1>・1ページ目...</h1>
</body></html>
```

# 文字を書き出す

**document.write(文字列)** [メソッド]






JavaScriptでブラウザに文字を書き出す時には、「write()」メソッドを使用します。その時に、JavaScript内にHTMLのタグを書けば、普通にHTMLで記述した時と同じようにタグが評価され、書き出させた文字は普通のテキスト文と同様に前後に挟まれたタグに従って表示されます。また、JavaScript自体も、HTMLのタグと同じように文字を修飾するコマンドを多数持っています。JavaScriptが持っている文字を修飾するメソッドに関しては、stringオブジェクト(P.545～)を参照してください。
Netscape Navigator 4.Xでは、ウィンドウサイズを変更した時に、「write()」メソッドで書き出された文字が消えてしまいます。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type"
content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*文字を書き出す
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

```
<p>

<script type="text/javascript">
<!--
document.write("<b>例1)こうしても</b>");
//-->
</script>
<br>
<b>
<script type="text/javascript">
<!--
document.write("例2)こうしても");
//-->
</script>
</b>
<br>
<script type="text/javascript">
<!--
document.write("例3)こうしても".bold());
//-->
</script>

<br>
結果は同じ。
</p>
</body>
</html>
```

bold()：「stringオブジェクト」の「太字(ボールド)にする」(P.547)



**重要**

### 書き出された文字が文字化けする時は

「document.write()」で文字を書き出したり、「alert()」メソッドなどのダイアログボックスに文字を表示した時に、「表示」といった漢字が「侮ｦ」と文字化けしてしまう場合があります。これは、日本語の文字コードと英語の文字コードとの間に問題があるためです。

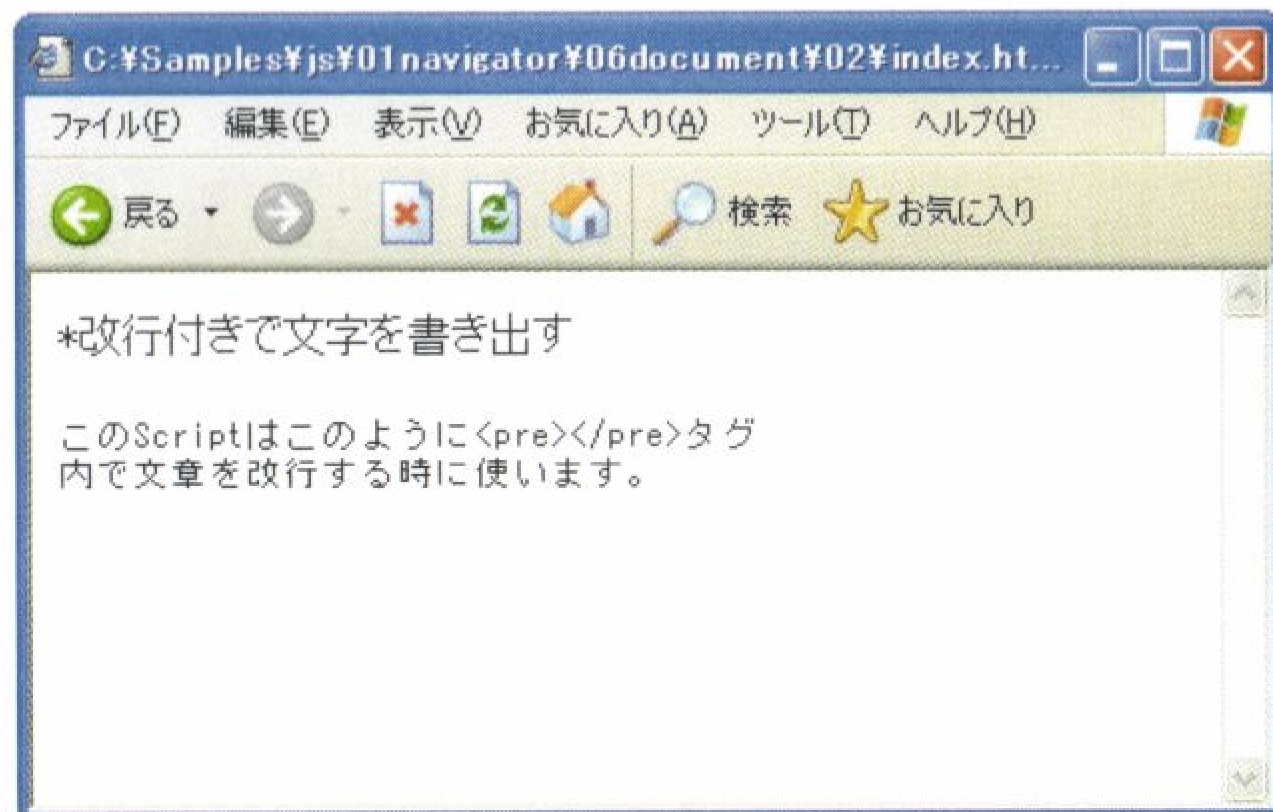
もし、文字化けが発生した場合、上のサンプルのように「表￥示」と、文字化けしている文字の後ろに「￥」(またはバックスラッシュ)を入れます。

この他にも、「可能￥」や「予￥定」や「申￥し訳」などのわりと使いそうな漢字も同様の問題があり、その結果エラーを起こす場合があります。

# 改行付きで文字を書き出す

**document.writeln(文字列)** [メソッド]






「writeln()」メソッドは、コマンドの終了位置に改行コードを付けて文字を書き出します。<pre>内でのみで意味を持ちます。

## Sample

```
<pre>
<script type="text/javascript">
<!--
document.writeln("このScriptはこのように&lt;pre&gt;&lt;/pre&gt;タグ");
document.writeln("内で文章を改行する時に使います。");
//-->
</script>
</pre>
```

<pre>：「スタイルとレイアウト」の「空白や改行をそのまま表示させる」(P.46)

### 重要

**長い文章を書き出したい時は**

「document.write()」などを使って長い文章を書き出した時に、ブラウザのバージョンによっては、エラーが発生する時があります。

これは、Netscape Navigatorが1行中に1バイト文字で255字まで、2バイト文字だとその半分の文字しか扱えなかったことからくる問題です。

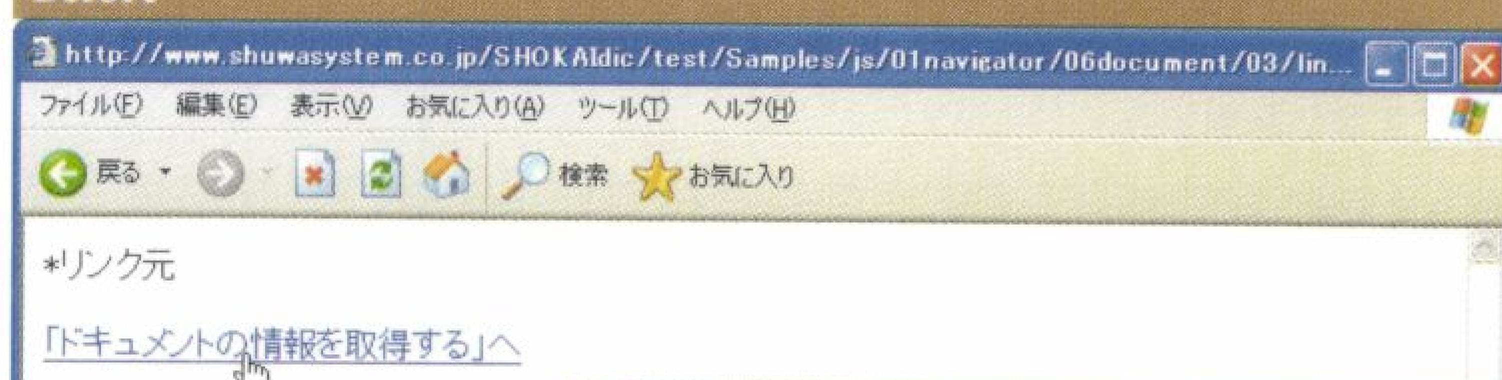
この問題を回避するために、「document.write("文字列A"+"文字列B")」といった具合に、JavaScriptで書き出す文章は短く分けることをお勧めします。

JavaScript

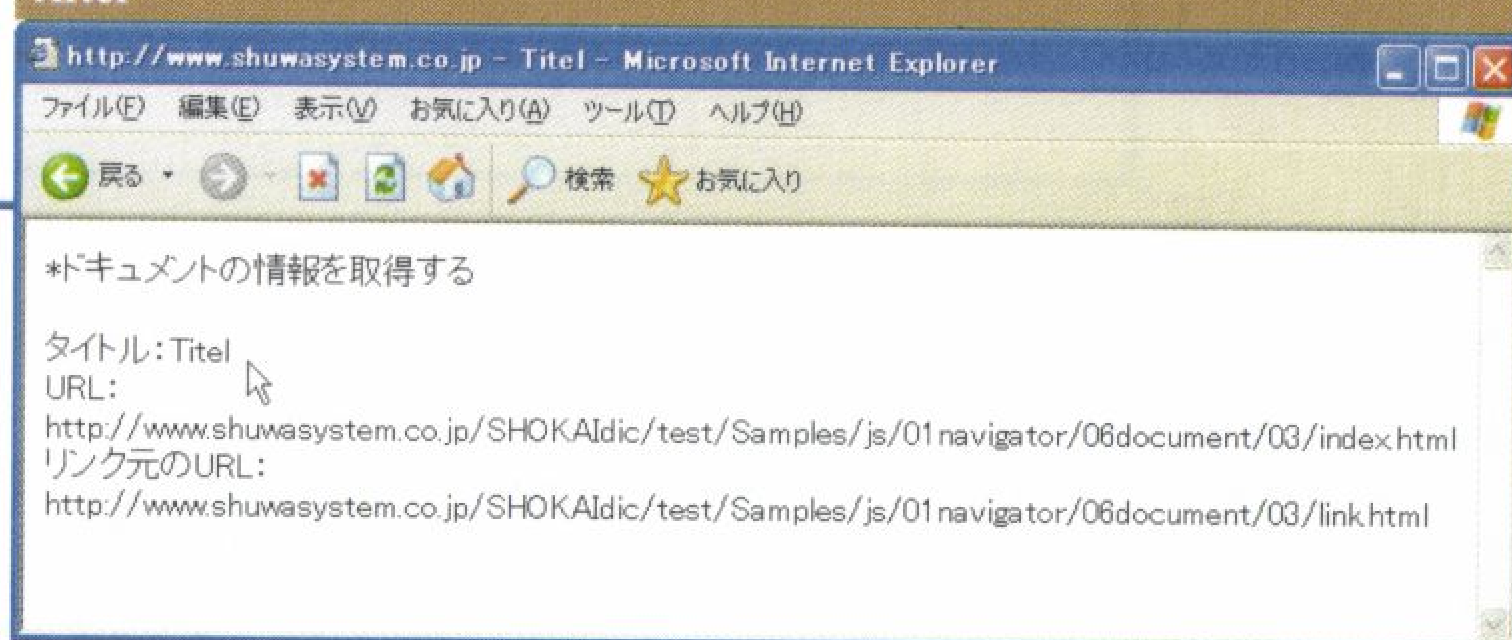
# ドキュメントの情報を取得する

| | |
|---|---|
| **document.title** | [プロパティ] |
| **document.URL** | [プロパティ] |
| **document.referrer** | [プロパティ] |

Before




After




documentオブジェクトが記述してあるHTMLファイルの情報を取得するプロパティです。「title」プロパティはHTMLの<title>部分の値を、「URL」プロパティはそのHTMLファイル自身のURLの値を、「referrer」プロパティはHTMLファイルがリンクされていたURLの値を、それぞれ持っています。

「referrer」プロパティは、Internet Explorer 3.Xでは、リンク元のURLではなく自分自身のURLを表示してしまいます。この問題は、Internet Explorer 4.XやMacintosh版のInternet Explorer 3.01では解消されています。しかし、Internet Explorer 5.0でもリンク元URLを取得できない場合がありますので、注意してください。

実際に試す場合には、「link.html」と一緒にサーバーにアップロードしてください。

これらのプロパティは読み出し専用です。

## Sample

```
<script type="text/javascript">
<!--
document.write("タイトル:",document.title);
document.write("<br>");
document.write("URL:",document.URL);
document.write("<br>");
document.write("リンク元のURL:",document.referrer);
//-->
</script>
```

# ファイルの更新日時を取得する

**document.lastModified** [プロパティ]






「lastModified」プロパティは、documentオブジェクトが記述してあるHTMLファイルの、最終更新日時を持っています。
この日時は、HTMLファイルが置かれているHTTPサーバーのタイムスタンプが参照されます。HTTPサーバーは国際標準時で運用されていることが多く、ファイルの最終更新日時が日本時間とずれる場合があります。
このプロパティは読み出し専用です。

## Sample

```
<script type="text/javascript">
<!--
document.write("Last update:",document.lastModified);
//-->
</script>
```

### 重要

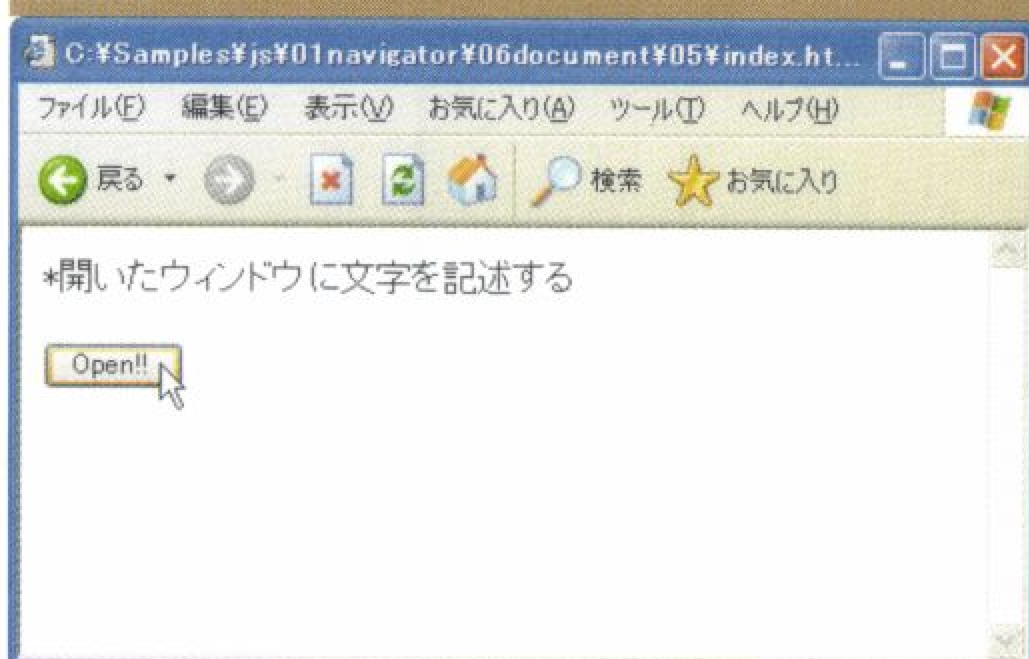
**MacintoshでHTTPサーバーを運用する時の注意**

「lastModified」プロパティは、どのバージョンのNetscape NavigatorもMacintoshからは正確なファイル更新日を取得することができません。Macintoshを使ってHTTPサーバーを運用している場合は、注意が必要です。

# 開いたウィンドウに文字を記述する

| | |
|---|---|
| document.write(文字列) | [メソッド] |
| document.open() | [メソッド] |
| document.close() | [メソッド] |

Before




After




サンプルでは、URLの指定なしに「window.open」を実行することにより、何も記述していないウィンドウを開き、その中にJavaScriptでドキュメントを記述しています。
ウィンドウへの記述方法は、まず「document.open()」でドキュメントストリームを開き、そこへ「document.write()」でドキュメントを書き出します。そして、ドキュメントの記述を終らせる時は、「document.close()」でドキュメントストリームを閉じます。
「document.write()」で書き出された文字は、HTMLタグやJavaScriptを通常通りに評価するので、画像を貼り込んだり、JavaScriptを記述したりできます。
しかし、Netscape Navigator 2.0では、画像ファイルやリンクのURLを絶対パスで指定しなければ画像ファイルやリンクが探せない時があり、その結果、画像が表示されなかったり、リンクが機能しない場合があります。そのような場合は、画像ファイルやリンクのURLを絶対パスで指定するようにしてください。
「document.open()」は省略可能ですが、「document.close()」は必ず記述して、明示的にドキュメントストリームを閉じるようにしてください。
もしも「document.close()」が記述されていない場合は、JavaScriptは、最後の「document.write()」が読み込まれても、ドキュメントがまだ後に続くものと判断して、待機状態のままになります。その結果、Netscape Navigatorでは最後の行が表示されなかったり、Internet Explorerで読み込み中を示す「e」のアイコンが回りっぱなしになり、最悪の場合は何も表示されないという状態になります。Netscape Navigatorの場合は、最後の行に<br>などの改行タグを入れることによって最後の行を表示させることが可能ですが、本質的な解決ではありません。

## Sample

```
<script type="text/javascript">
<!--
function DWO1(){
    var DW1;
    DW1=window.open("","DocWin1");
    DW1.document.open();
    DW1.document.write("<html><head><title>DOCUMENT_SAMPLE</title>
");
    DW1.document.write("<"+"script language='JavaScript'>");
    DW1.document.write("function WClose2() {window.close()}");
    DW1.document.write("</script>");
    DW1.document.write("</head>");
    DW1.document.write("<body>");
    DW1.document.write("<img name='kao' src='image.gif' alt='image
.gif' width='100' height='100'>");
    DW1.document.write("<br>");
    DW1.document.write("イメージも表¥示¥できます。");
    DW1.document.write("<p>");
    DW1.document.write("<hr>");
    DW1.document.write("<form>");
    DW1.document.write("<input type='button' name= 'Wcl2' value='
Window Close ' onClick='WClose2()'>");
    DW1.document.write("</form>");
    DW1.document.write("<br>");
    DW1.document.write("JavaScriptも記述できます。");
    DW1.document.write("</body>");
    DW1.document.write("</html>");
    DW1.document.close();
}
//-->
</script>
  ～中略～
<body>
＊開いたウィンドウに文字を記述する
<p>
<form>
<input type="button" value=" Open!! " onClick="DWO1()">
</form>
</p>
</body>
```


window.open()：「windowオブジェクト」の「新しいウィンドウを開く」(P.388)
window.close()：「windowオブジェクト」の「ウィンドウを閉じる」(P.341)

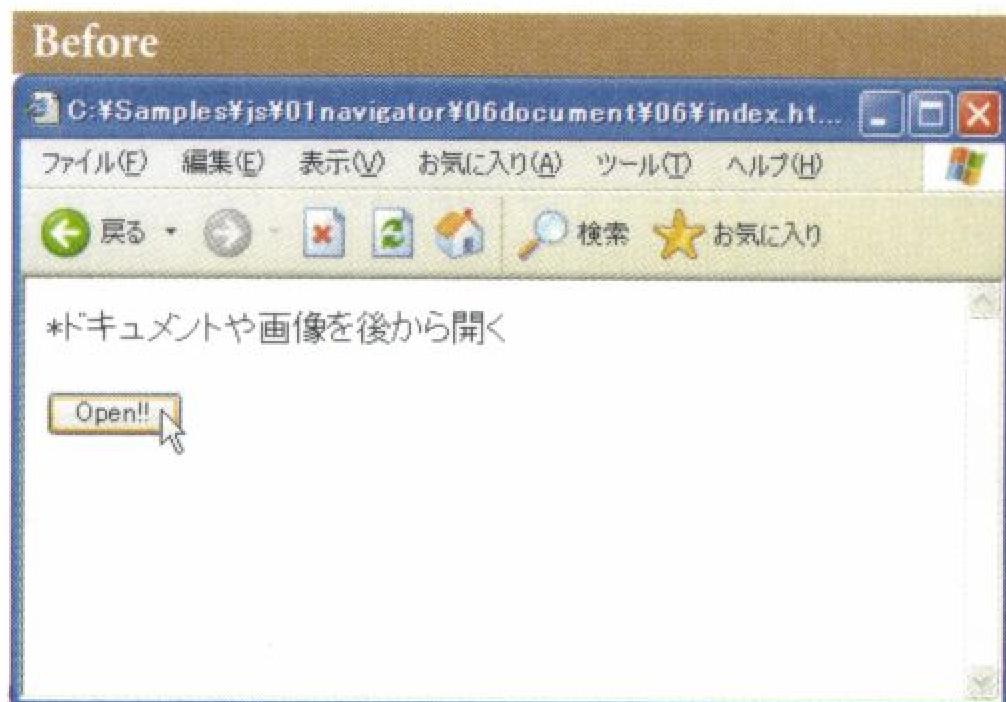
IE6.0 IE5.5 IE5.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac

JavaScript

# ドキュメントや画像を後から開く

**document.write(*文字列*)** [メソッド]
**document.open()** [メソッド]
**document.close()** [メソッド]

Before




After




新しいウィンドウに文字を記述するのと同じ要領で、ブラウザに表示されている文字を書き換えることができます。
上の例では、[Open!!]ボタンがクリックされると、「document.open()」でドキュメントストリームが開かれます。同時に、現在表示されている文字やフォームなどが消され、その後に「document.write()」で文字が書き出されます。
ウィンドウに文字を記述した時と同様に、「document.close()」は必ず入れるようにしてください。
Safariでは、最後に「<br>」を入れないと、最終行が表示されない場合があります。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type"
content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function Doc4(){
    document.open();
    document.write("<p> ");
    document.write("<img src='image.jpg' alt='image.jpg' width='10
0' height='100'>");
    document.write("</p> ");
    document.write("このようにイメージを開くこともできます。<br>");
    document.close();
```

```
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
＊ドキュメントや画像を後から開く
<p>
<form>
<input type="button" value=" Open!! " onClick="Doc4()">
</form>
</p>
</body>
</html>
```

**注意**

## 「window.open()」で開いたウィンドウに「document.write()」で文字を書き出す時の注意

「開いたウィンドウに文字を記述する」(P.388)のように、「window.open()」で新しく開いたウィンドウに「document.write()」で文字を書き出すスクリプトは、ウィンドウを開いてHTMLファイルを読み込むスクリプトに比べて、外部からファイルを読み込む必要がないため表示が早くなります。ちょっとしたメッセージを表示する場合には便利なのですが、このスクリプトは、Internet Explorerではうまく機能しない時があります。特に、「開いたウィンドウに文字を記述する」のサンプルのように、書き出す文字にJavaScriptのスクリプトが混じっているような時などは、エラーになる場合が多くあります。また、同じバージョンのInternet Explorerでも、新しいものではエラーが出なくても、より古いものではエラーになる場合があります。

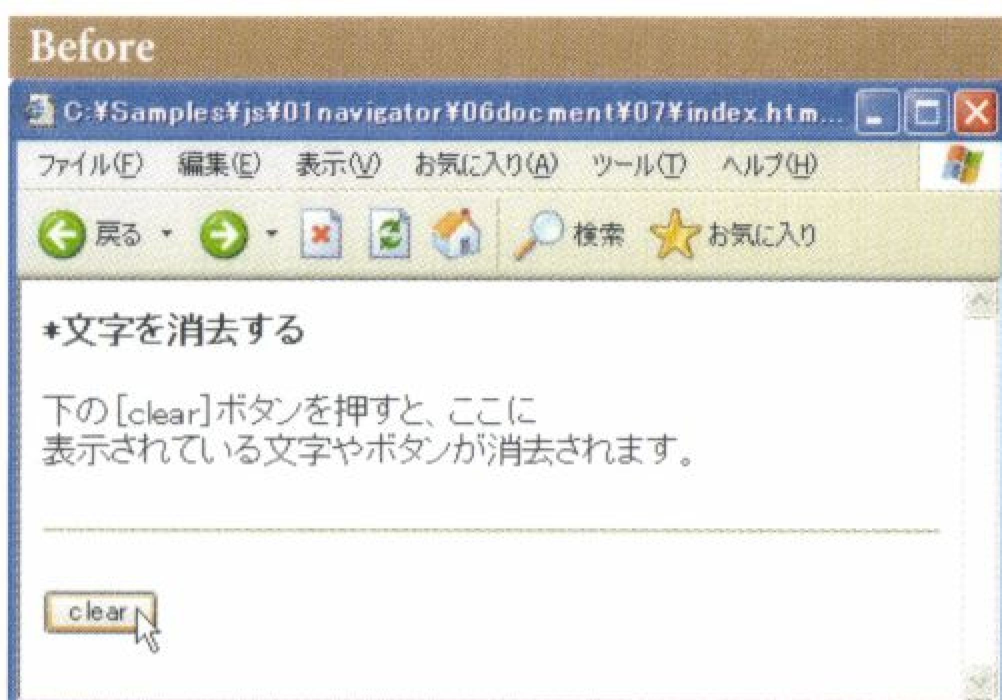
本書の「開いたウィンドウに文字を記述する」などのサンプルでは、チェックの結果、一部のInternet Explorerでは不安定な部分がありました。そのため、「window.open()」で新しく開いたウィンドウに「document.write()」で文字を書き出すスクリプトは、基本的にInternet Explorer 4.0には対応していないことにしていますが、書き出す内容や環境によっては正常に作動します。反対に言えば、一見自分の環境では正常に動作するように見えても、他の環境ではエラーになる場合がありますので、十分にテストを行うようにしてください。

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

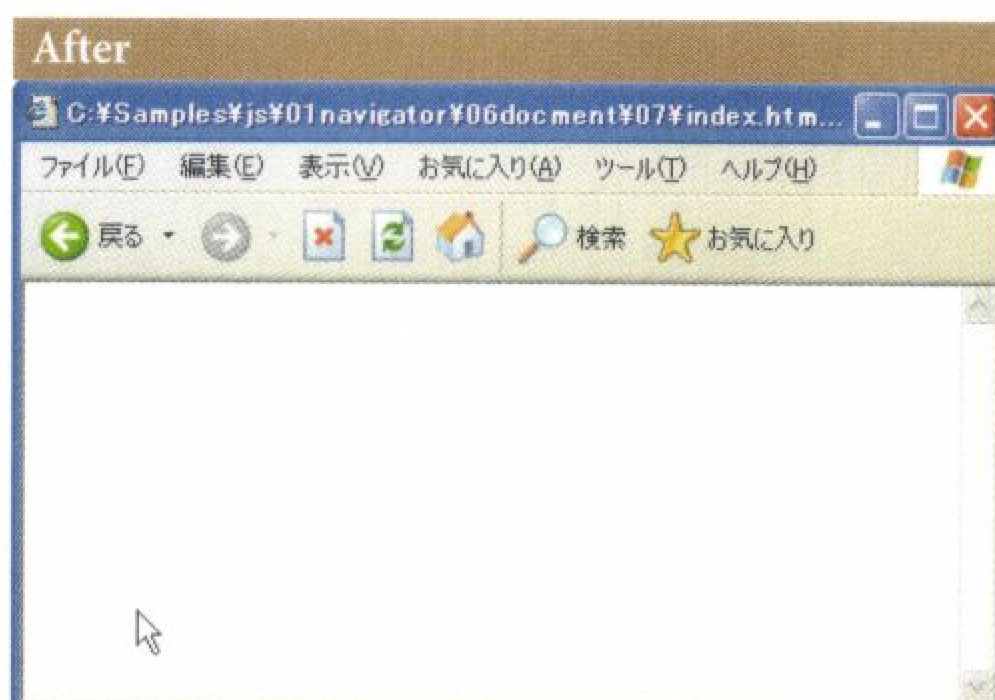
# 文字を消去する

**document.open()** [メソッド]

Before




After




「document.open()」でドキュメントストリームが開いた時、ブラウザに表示されている文字などが消されることを利用したスクリプトです。
[clear]ボタンが押されると「document.open()」が実行され、表示されている文字が消去されます。
JavaScript1.0には、ドキュメントを消去するために「clear()」というメソッドが用意されていましたが、JavaScript1.1以降では、「clear()」メソッドは削除されています。
このスクリプトは、Macintosh版のInternet Explorer 4.Xでは動作しない場合があります。

## Sample

```
<html><head>
<mata http-equiv="Content-Script-Type" content="text/javascript">
<title></title>
<script type="text/javascript">
<!--
function Dclear(){ document.open() }
//-->
</script>
  ～中略～
</head>
<body>
<b>*文字を消去する</b>
<p>
下の[clear]ボタンを押すと、ここに
<br>
表示されている文字やボタンが消去されます。
</p>
<hr>
<form>
<input type="button" value=" clear " onClick="Dclear()">
</form>
</body></html>
```

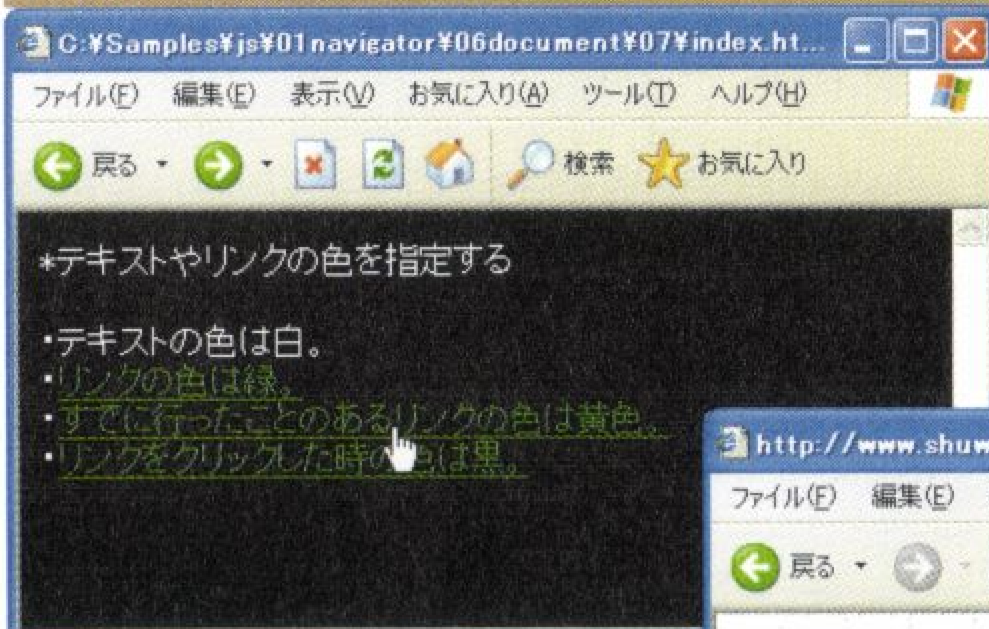
ドキュメントを操作する

# テキストやリンクの色を指定する

**document.alinkColor="*色指定*"** [プロパティ]
**document.bgColor="*色指定*"** [プロパティ]
**document.fgColor="*色指定*"** [プロパティ]
**document.linkColor="*色指定*"** [プロパティ]
**document.vlinkColor="*色指定*"** [プロパティ]



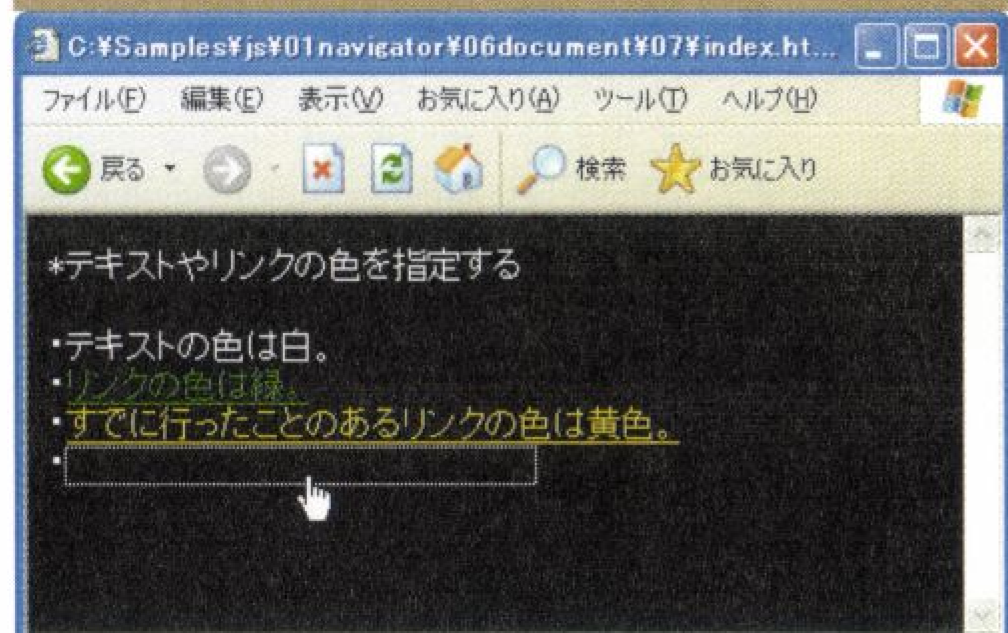
Before







After




「alinkColor」プロパティはアクティブリンクの色の値を、「fgColor」プロパティはフォアグラウンド、つまりテキストの色の値を、「linkColor」プロパティはリンクの色の値を、「vlinkColor」プロパティはすでに行ったことのあるリンクの色の値を、それぞれ持っています。
サンプルでは、それぞれのプロパティに色の値を設定することによって、ブラウザに表示されるテキストやリンクの色を指定しています。
色指定は、色の名前か16進数で設定します。
これらの設定は、<body>内に記述するHTMLでの色指定より、優先されます。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<mata http-equiv="Content-Script-Type" content="text/javascript">
<title></title>

<script type="text/javascript">
<!--
document.fgColor="white";
document.linkColor="green";
document.vlinkColor="#ffff00";
document.alinkColor="#000000";
//-->
</script>

</head>
<body bgcolor="#000000">
<p>*テキストやリンクの色を指定する</p>
・テキストの色は白。<br>
・<a href="NoLink.html">リンクの色は緑。</a>
<br>
・<a href="http://www.shuwasystem.co.jp/">すでに行ったことのあるリンクの色
は黄色。</a>
<br>
・<a href="#">リンクをクリックした時の色は黒。</a>
</body>
</html>
```

<body bgcolor="～">：「基本的な内容」の「全体の背景色を設定する」(P.13)
色指定：巻末付録「カラーチャート1～3」(巻末)

## 注意

### Netscape Navigator 2.0でドキュメントをその場で書き換えるスクリプトを使用する時の注意

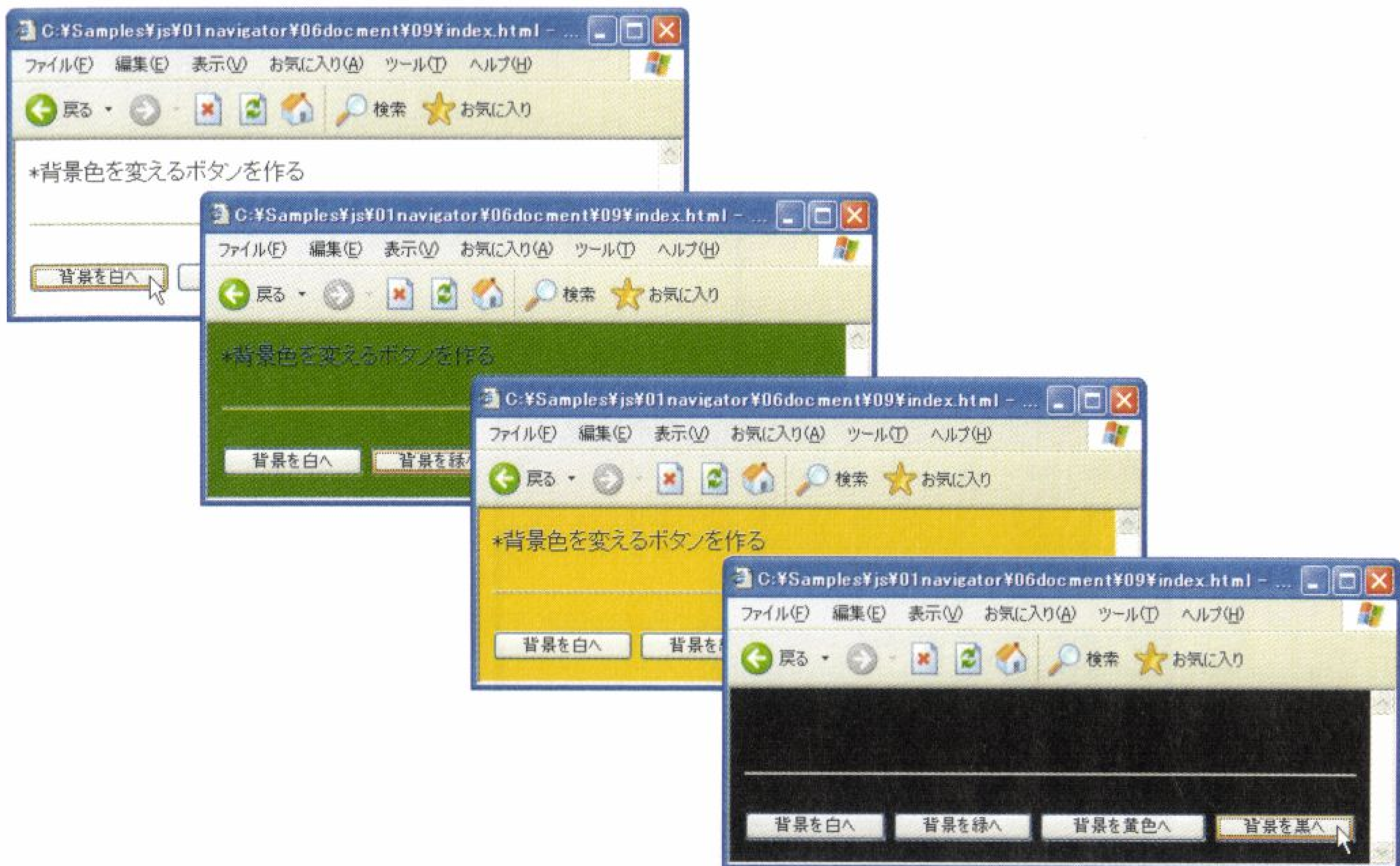
Netscape Navigator 2.0では、ドキュメントや画像を後から開くスクリプトや、文字を消去するスクリプトは非常に不安定です。最悪の場合はシステムエラーを引き起こす場合があります(Macintosh版で確認)。

また、「document.close()」を記述すると、本来そこでドキュメントの記述が終了したと判断されなければいけないはずなのですが、反対に待機状態になってしまう場合があります。

# 背景色を変えるボタンを作る

**document.bgColor="色指定"** [プロパティ]






「bgColor」プロパティは、バックグラウンドの色の値を持っています。
サンプルでは、ボタンをクリックした時に、「document.bgColor」プロパティで設定している色の値が評価され、背景色がその場で変わります。

## Sample

```
<body bgcolor="#ffffff">
<p>*背景色を変えるボタンを作る</p>
<hr>
<form>
 <input type="button" value=" 背景を白へ " onClick="document.bgColor
='white'">
 <input type="button" value=" 背景を緑へ " onClick="document.bgColor
='green'">
 <input type="button" value=" 背景を黄色へ " onClick="document.bgColo
r='#ffff00'">
 <input type="button" value=" 背景を黒へ " onClick="document.bgColor
='#000000'">
</form>
</body>
```

色指定:巻末付録「カラーチャート1～3」(巻末)

# テキストの色を変えるボタンを作る

**document.fgColor="色指定"** [プロパティ]




Internet ExplorerやDOMが採用されたNetscape 6.X以降では、ページのレイアウトが確定した後からでも、表示されているテキストの色を変えることができます。
サンプルでは、ボタンをクリックされた時、ブラウザのテキストの色の値を持った「document.fgColor」プロパティで設定している色が評価され、テキストの色がその場で変わります。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<title></title>
</head>
<body bgcolor="#ffffff">
*テキストの色を変えるボタンを作る
<p>
Internet ExplorerやMozillaでは、レイアウト確定後も文字の色を変えることができます。
</p>
<hr>

<form>
 <input type="button" value="文字を白へ" onClick="document.fgColor='white'">
 <input type="button" value="文字を緑へ" onClick="document.fgColor='green'">
 <input type="button" value="文字を黄色へ" onClick="document.fgColor='#ffff00'">
 <input type="button" value="文字を黒へ" onClick="document.fgColor='#000000'">
</form>

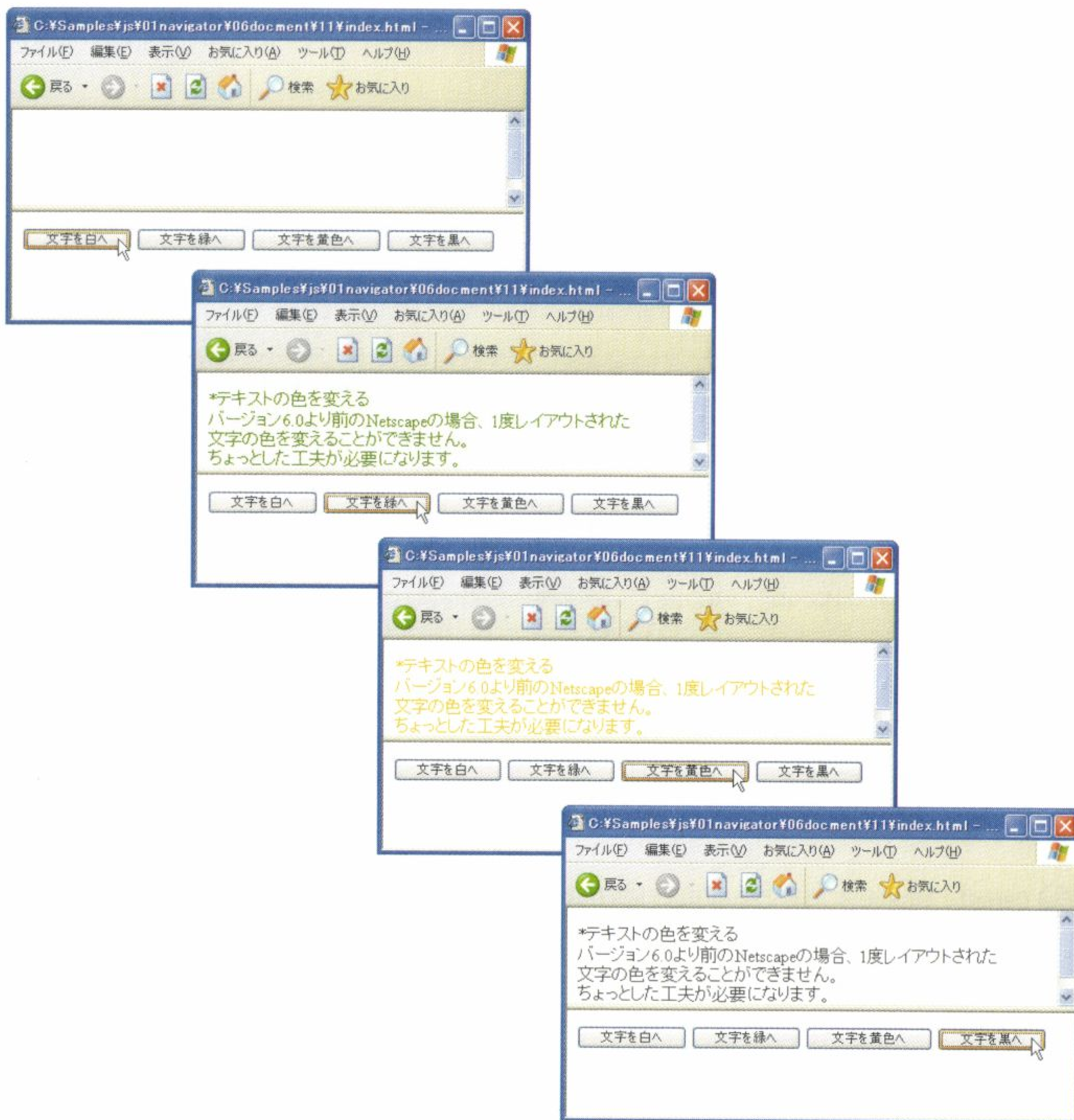
</body>
</html>
```


parent：「frameオブジェクト」の「入力されたURLを別フレームに表示する」(P.374)
付録コラム「DynamicHTMLとは」(P.652)
色指定:巻末付録「カラーチャート1～3」(巻末)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X Safari IE5-mac IE4-mac

# テキストの色を変える

**document.write(文字列)** [メソッド]




バージョン6.0より前のNetscape Navigatorでは、1度レイアウトが確定してしまうと、バックグラウンドの色以外のテキストやリンクの色を変更することができません。サンプルでは、フレーム「f2」からボタンがクリックされた時に、テキストの色の値と一緒にフレーム「f1」へ、フレーム「f1」と同じレイアウトのドキュメントを書き出しています。実際に試す場合には、この他に「f1.html」を用意してください。

## Sample

**【フレームウィンドウ】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" >
<html>
```

```
<head>
<title></title>
</head>
<frameset rows="*,100">
    <frame src="f1.html" name="f1">
    <frame src="f2.html" name="f2">
</frameset>
<noframes>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</noframes>
</html>
```

**【f2.html】**

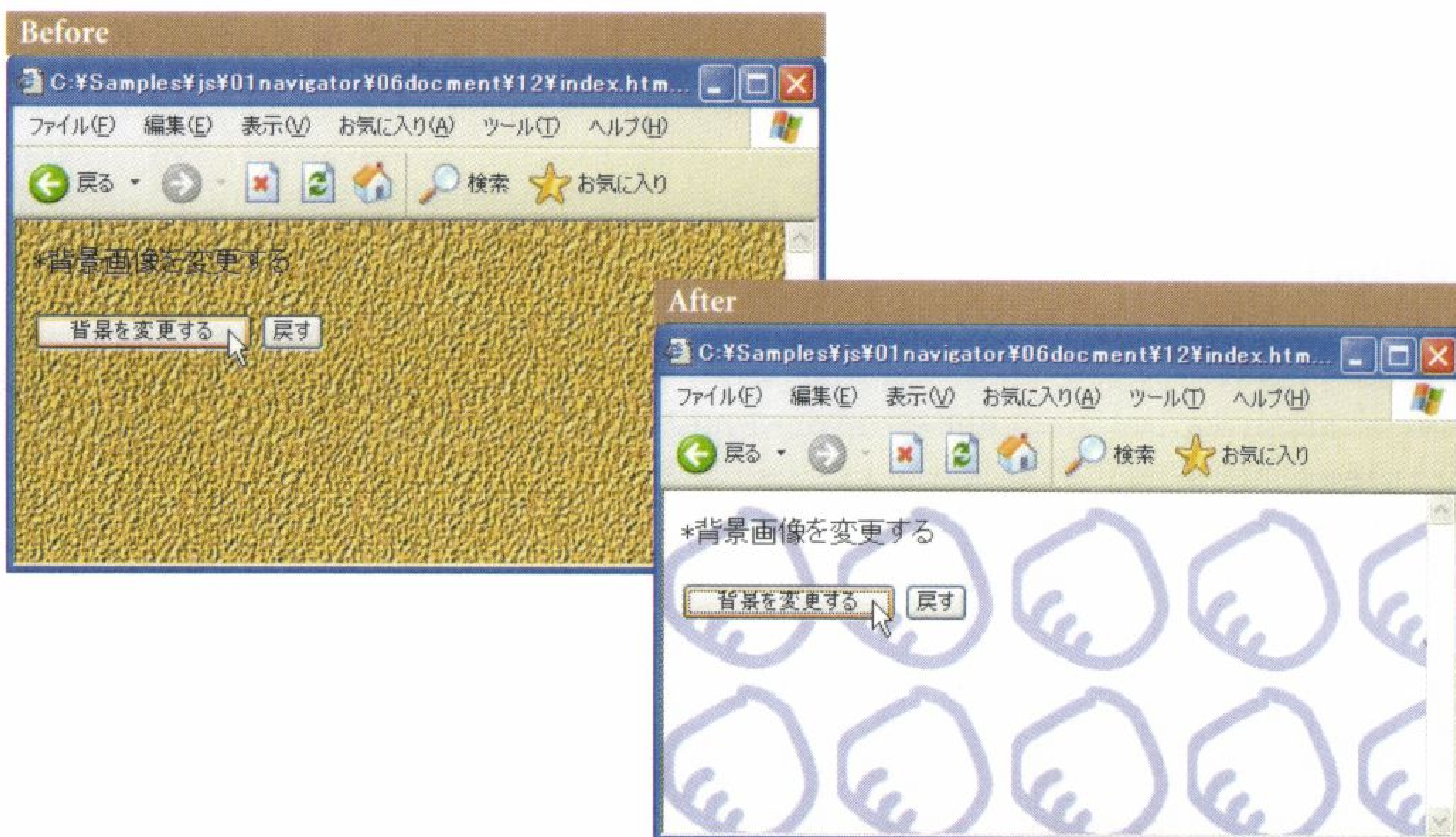
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<title>
</title>
<script type="text/javascript">
<!--
function TP(TC) {
    parent.f1.document.open();
    parent.f1.document.write("<body bgcolor='#ffffff' text=");
    parent.f1.document.write(TC);
    parent.f1.document.write(">");
    parent.f1.document.write("*テキストの色を変える<br>");
    parent.f1.document.write("バージョン6.0より前の");
    parent.f1.document.write("Netscapeの場合、1度レイアウトされた<br>");
    parent.f1.document.write("文字の色を変えることができません。<br>");
    parent.f1.document.write("ちょっとした工夫が必要になります。<br>");
    parent.f1.document.close()
}
//-->
</script>
</head>
<body bgcolor="#ffffff">
<form>
 <input type="button" value=" 文字を白へ " onClick="TP('white')">
 <input type="button" value=" 文字を緑へ " onClick="TP('green')">
 <input type="button" value=" 文字を黄色へ " onClick="TP('#ffff00')">
 <input type="button" value=" 文字を黒へ " onClick="TP('#000000')">
</form>
</body>
</html>
```

 parent：「frameオブジェクト」の「入力されたURLを別フレームに表示する」(P.374)
色指定：巻末付録「カラーチャート1～3」(巻末)

# 背景画像を変更する

**document.getElementById(オブジェクト名).background=*URL***

[プロパティ]




Netscape 6.0からは、DOMの採用により、基本的にHTMLタグのあらゆる属性へ、JavaScriptからアクセスできるようになりました。
サンプルでは、それを利用して<body>の「background」の値を変更することによって、ブラウザの背景画像を変更しています。
サンプルでは、Internet Explorer 4.Xとそれより後のバージョンのInternet ExplorerのDOMの実装の違いを考慮し、「getElementById()」メソッドと「all()」メソッドのどちらをサポートしたブラウザでも、対応できるようにしています。また、Internet Explorer 5.1では、背景画像を変更する時に、ブラウザに表示されている文字が背景画像に塗りつぶされてしまったようになる場合があります。
JavaScript1.5で追加されたプロパティです。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<title></title>

<script type="text/javascript">
<!--
function BackImg1(){
```

```
    if(document.getElementById){
        document.getElementById("BackImg").background = "BACK2.jpg"
    }
    if(document.all){
        document.all("BackImg").background = "BACK2.jpg"
    }
}
function BackImg2(){
    if(document.getElementById){
        document.getElementById("BackImg").background = "BACK1.jpg"
    }
    if(document.all){
        document.all("BackImg").background = "BACK1.jpg"
    }
}
//-->
</script>

</head>
<body id="BackImg" background="BACK1.jpg">
＊背景画像を変更する
<p>
<form>
    <input type="button" value="背景を変更する" onClick="BackImg1('BA
CK2.jpg')">
    <input type="button" value="戻す" onClick="BackImg2('BACK1.jpg
')">
</form>
</P>
</body>
</html>
```

付録コラム「DynamicHTMLとは」(P.652)

# クッキーファイルへ書き込む

## document.cookie [プロパティ]




JavaScriptでは、cookieファイルに情報を書き込み、利用することができます。
cookieファイルは、ローカルディスク上に置かれ、個人を特定する情報や特定サイトを利用するのに必要な情報を保存しているので、そのユーザーがサイトを訪れた回数を表示したり、ユーザーに合わせた情報を表示することができます。
JavaScriptは、セキュリティ上の関係から、ローカルファイルなどのコンピュータのローカル資源へのアクセスは、基本的に行えないようになっています。cookieファイルは、JavaScriptが唯一アクセスできるローカル資源です。
cookieファイルの使用については、記録できる情報量を制限するなどの対策がとられているのですが、CGIなどで利用した場合はユーザーの行動をある程度監視することができるなどの理由から嫌う人もいます。
サンプルでは、HTMLファイルが読み込まれた時のローカルタイムをcookieファイルへ書き込み、それをまたcookieファイルから読み出して表示しています。
Macintosh版のInternet Explorerでは、設定にかかわらずcookieの値を取得できない場合があります。

### Sample

```
<script type="text/javascript">
<!--
Time = new Date();
document.cookie=Time.toLocaleString();
document.write("AccessTime:"+document.cookie);
//-->
</script>
```

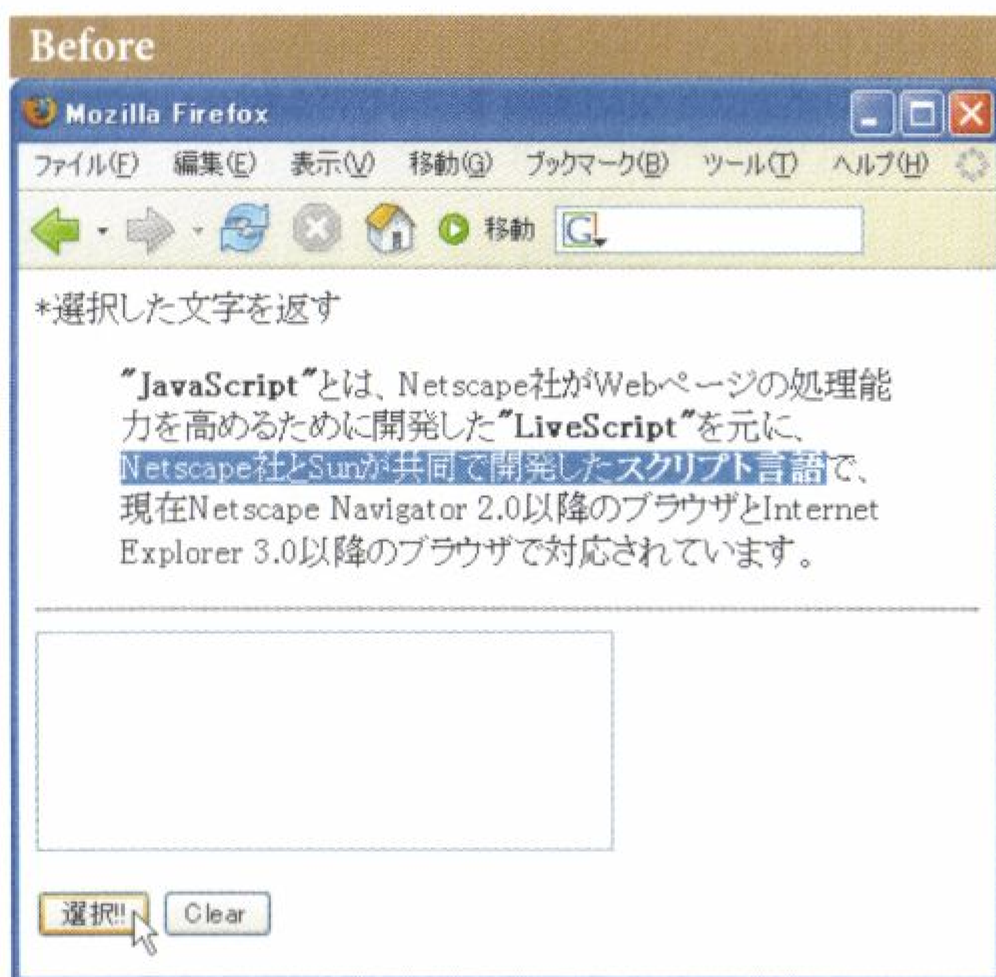

toLocaleString():「Dateオブジェクト」の「国際標準時やローカルタイムを表示する」(P.506)

# 選択した文字を返す

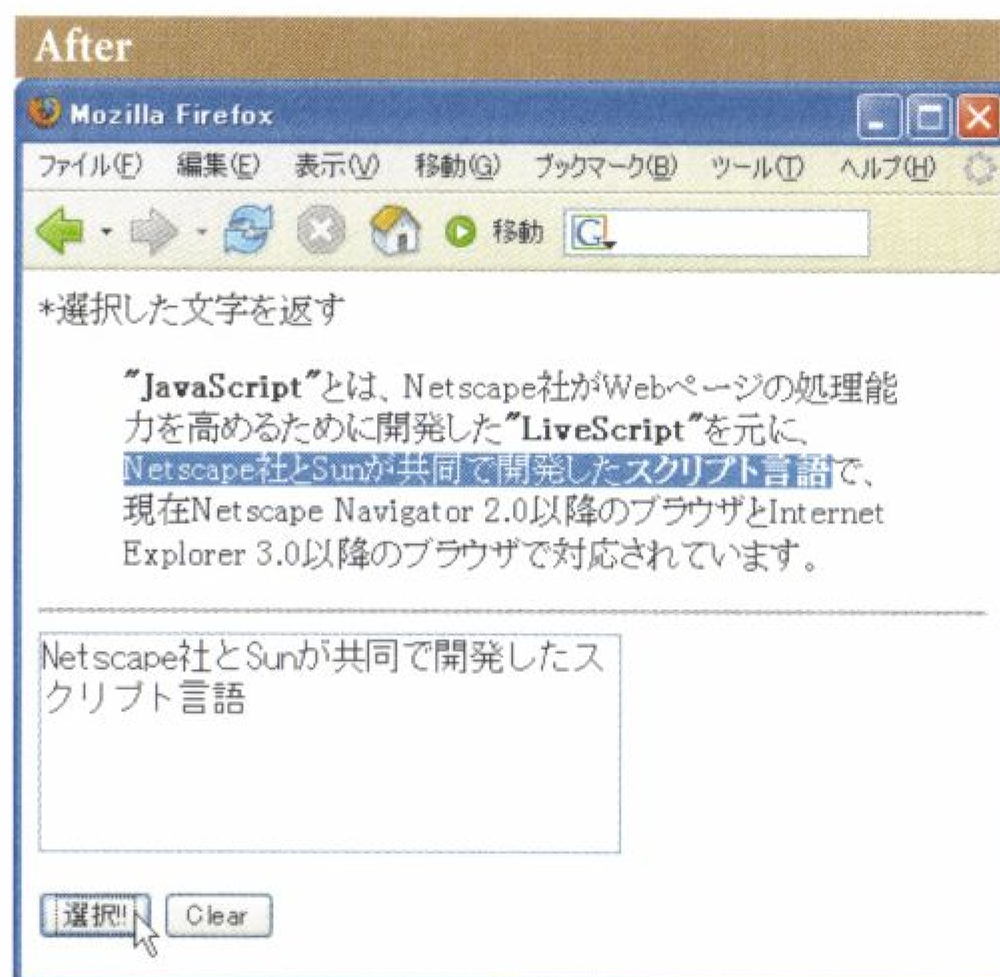
## document.getSelection() [メソッド]



Before




After




「getSelection()」メソッドは、マウスなどで選択された文字を返します。
フォームに値を渡すことが可能なので、サンプルのように選択した文字をテキストエリアに書き出すこともできます。
JavaScript1.2で追加されたメソッドです。

### Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function selct() {
    document.OUTP.outp.value = document.getSelection();
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
```

```
<body>
＊選択した文字を返す
<blockquote>
<b>"JavaScript"</b>とは、Netscape社がWebページの処理能力を高めるために開発した<b>"LiveScript"</b>を元に、Netscape社とSunが共同で開発した<b>スクリプト言語</b>で、現在Netscape Navigator 2.0以降のブラウザとInternet Explorer 3.0以降のブラウザで対応されています。
</blockquote>
<hr>
<form name="OUTP">
<textarea name="outp" rows=5 cols=30>
</textarea>
<p>
<input type="button" value=" 選択!! " onClick="selct()">
<input type="reset" value=" Clear ">
</p>
</form>
</body>
</html>
```

JavaScript

# その他のdocumentオブジェクト

## domain [プロパティ]

複数のサーバー名を、ひとつのドメイン名として取り扱えます。
たとえば、ひとつのウィンドウの別々のフレームに、「search.hamba.com」と「www.hamba.com」のふたつのサーバーからのデータを表示させる場合、「domain」プロパティによって各々のサーバ名を「hamba.com」として扱うことができます。
ただし、ドメイン名部分を変えることはできません。つまり「search.hamba.com」を「search.com」のように変えることはできません。また、1度ドメイン名を変更すると、変更前のドメイン名を参照できなくなります。
JavaScript1.1で追加されたプロパティですが、Internet Explorerは未対応です。
そのウィンドウが、他のウィンドウ、あるいはフレームから、プロパティなどのデータの参照を許している状態(taintingの状態)かどうかを調べます。
実際の指定は、次の例のように行います。

**Sample**

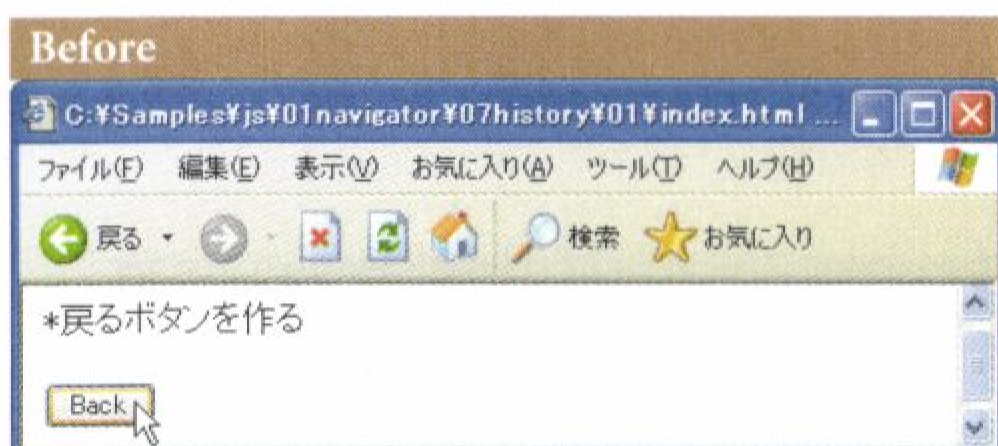
```
document.domain="指定するドメイン名"
```

JavaScript

# 戻るボタンを作る

## history.back() [メソッド]

Before




After




ブラウザの[戻る(Back)]ボタンと同じ働きをするスクリプトです。
フォームのボタンが押された時に、「onClick」で指定している「history.back()」が評価され、ひとつ前のページへ戻ります。戻るページが来歴内にない場合は何も起きません。
複数のページからリンクを張っているページで使うと効果的です。

### Sample

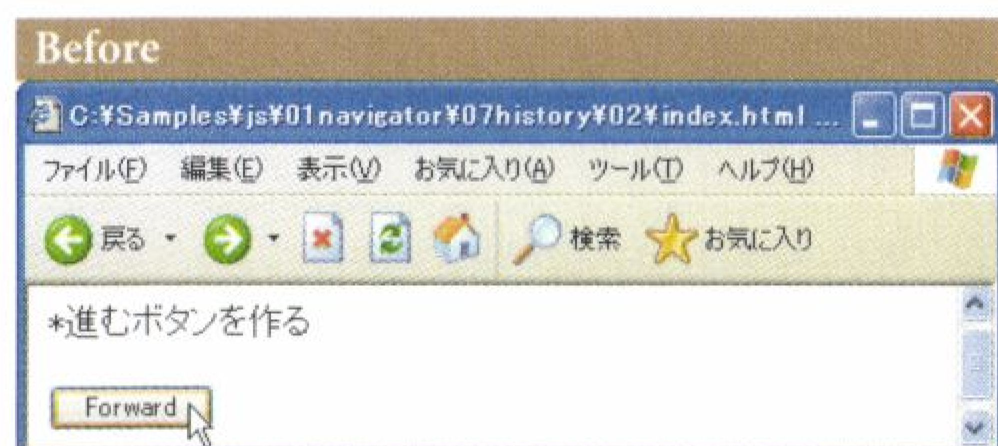
```
<form>
<input type="button" value=" Back " onClick="history.back()">
</form>
```

JavaScript

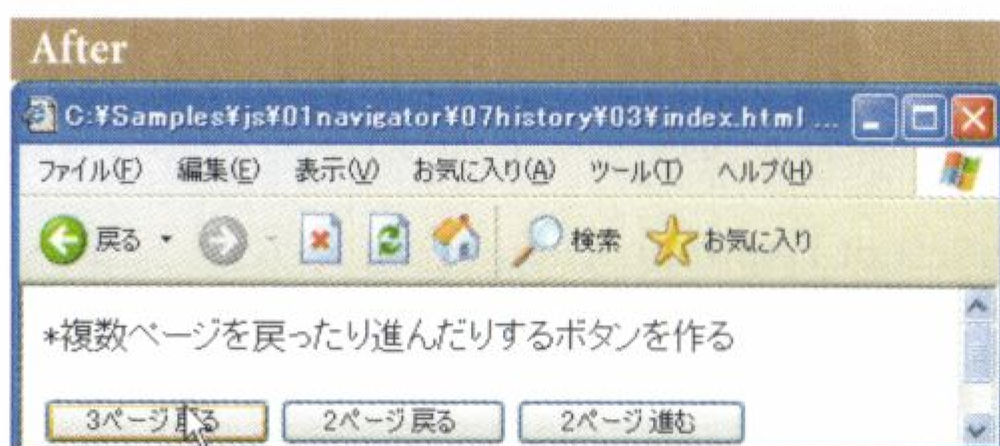
# 進むボタンを作る

## history.forward() [メソッド]

Before




After




ブラウザの[進む(Forward)]ボタンと同じ働きをするスクリプトです。
フォームのボタンが押された時に、イベントハンドラ「onClick」で指定している「history.forward()」が評価され、ひとつ先のページへ進みます。進むページが来歴内にない場合は何も起きません。

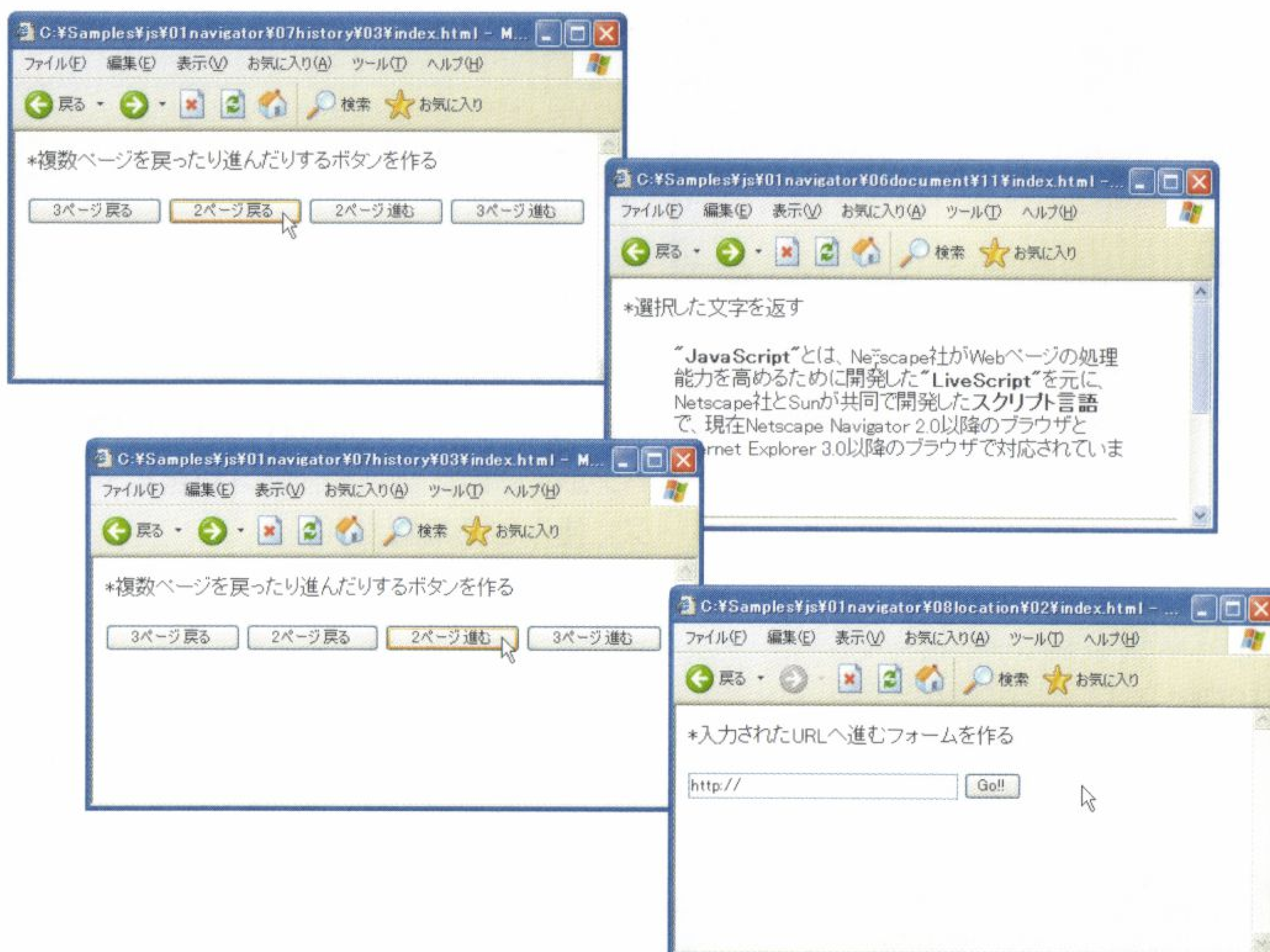
### Sample

```
<form>
<input type="button" value=" Forward " onClick="history.forward()">
</form>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 複数ページを戻ったり進んだりするボタンを作る

## onClick="history.go(*n*)" [メソッド]




複数のページを戻ったり進んだりするボタンのスクリプトです。
フォームのボタンが押された時に、イベントハンドラ「onClick」で指定している「history.go(n)」が評価され、「n」の数値分ページを移動します。
また、「go(URL)」と指定すると、来歴内の指定されたページを表示します。移動するページが来歴内にない場合は何も起きません。
Internet Explorer では、うまく機能しない場合があります。

### Sample

```
<form>
<input type="button" value=" 3ページ戻る " onClick="history.go(-3)">
<input type="button" value=" 2ページ戻る " onClick="history.go(-2)">
<input type="button" value=" 2ページ進む " onClick="history.go(2)">
<input type="button" value=" 3ページ進む " onClick="history.go(3)">
</form>
```

onClick：リファレンス「イベントハンドラ」の「onClick」(P.620)

# その他のhistoryオブジェクト

## current [プロパティ]

「current」プロパティは、現在表示されているウィンドウと同じURLの値を持っています。そのウィンドウが、他のウィンドウ、あるいはフレームからプロパティの参照を許している状態(taintingの状態)の時のみ機能します。
サンプルでは、「current」プロパティに「hamba.com」という値が含まれているかどうかを検索し、含まれている場合は「処理」を実行します。
JavaScript1.1で追加されたプロパティです。

### 用法

```
if (history.current.indexOf("hamba.com") != -1) { 処理 }
```

## previous [プロパティ]

「previous」プロパティは、ブラウザの[戻る(Back)]ボタンで得られるのと同じURLの値を持っています。
そのウィンドウが、他のウィンドウ、あるいはフレームからプロパティの参照を許している状態(taintingの状態)の時のみ機能します。
次の例では、「previous」プロパティに「hamba.com」という値が含まれているかどうかを検索し、含まれている場合は「処理」を実行します。
JavaScript1.1で追加されたプロパティです。

### 用法

```
if (history.previous.indexOf("hamba.com") != -1) { 処理 }
```

## next [プロパティ]

「next」プロパティは、ブラウザの[進む(Forward)]ボタンで得られるのと同じURLの値を持っています。
そのウィンドウが、他のウィンドウ、あるいはフレームからプロパティの参照を許している状態(taintingの状態)の時のみ機能します。
次の例では、「next」プロパティに「hamba.com」という値が含まれているかどうかを検索し、含まれている場合は「処理」を実行します。
JavaScript1.1で追加されたプロパティです。

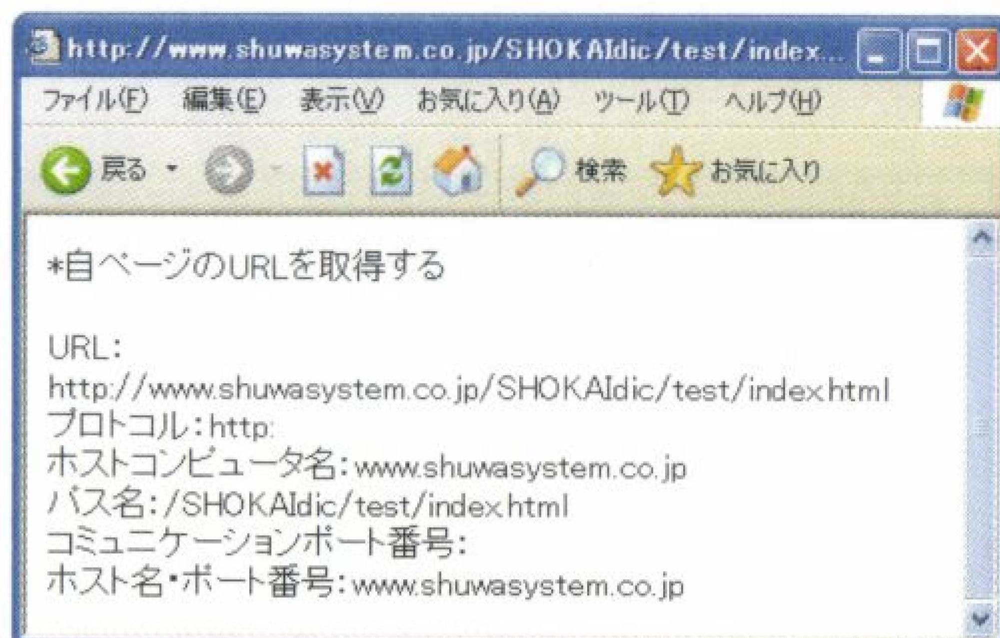
### 用法

```
if (history.next.indexOf("hamba.com") != -1) { 処理 }
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 自ページのURLを取得する

**location.href** [プロパティ]
**location.protocol** [プロパティ]
**location.hostname** [プロパティ]
**location.pathname** [プロパティ]
**location.port** [プロパティ]
**location.host** [プロパティ]




locationオブジェクトには、URLに関する情報が格納されています。
「href」プロパティはURL全体の値を、「protocol」プロパティはURL内のhttpやftpなどのプロトコル部分の値を、「hostname」プロパティはURL内のホスト名部分の値を、「pathname」プロパティはURL内のパス名部分の値を、「port」はURL内の:8080などのポート番号の値を、「host」はホスト名とポート番号部分の値を、それぞれ持っています。
locationオブジェクトには、これ以外にも「hash」プロパティ(アンカー)や「search」プロパティ(?で始まる問い合わせ文字列)があります。

## Sample

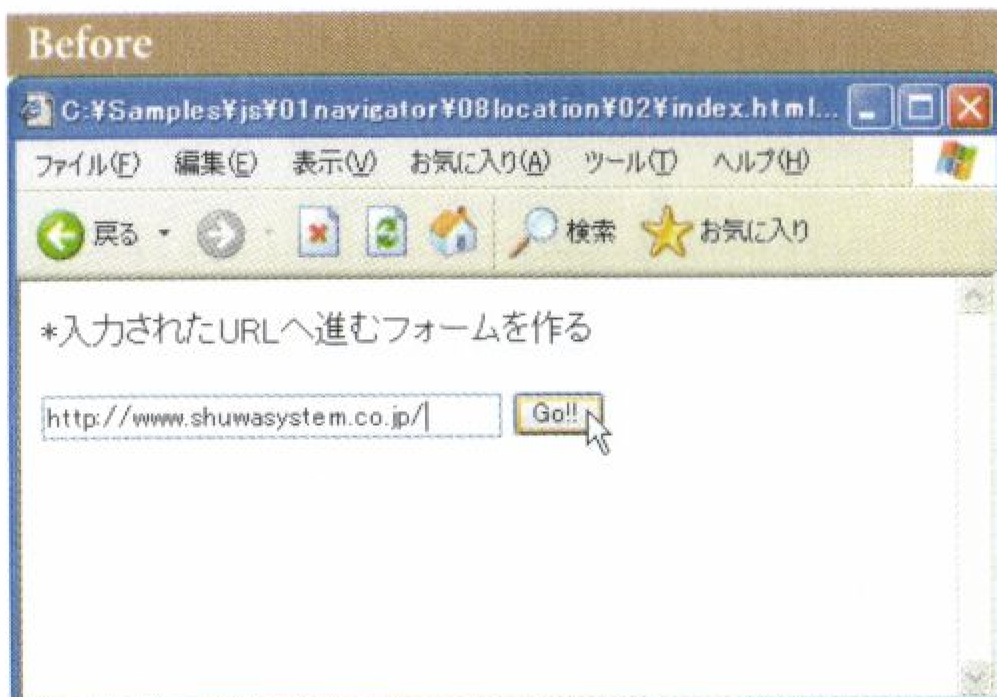
```
<script type="text/javascript">
<!--
document.write("URL:",location.href);
document.write("<br>");
document.write("プロトコル:",location.protocol);
document.write("<br>");
document.write("ホストコンピュータ名:",location.hostname);
document.write("<br>");
document.write("パス名:",location.pathname);
document.write("<br>");
document.write("コミュニケーションポート番号:",location.port);
document.write("<br>");
document.write("ホスト名・ポート番号:",location.host);
//-->
</script>
```

# 入力されたURLへ進むフォームを作る

## location.href ［プロパティ］



Before




After




サンプルでは、locationオブジェクトが動的にURLを変更できることを利用して、ボタンがクリックされたタイミングでフォームの内容を参照し、フォームに入力されたURLの値を「href」プロパティに設定することによって、フォームに入力されたURLがブラウザにロードされます。
また、もしフォームに何も入力されていない時には、警告用のダイアログボックスが開きます。入力されたURLが不正な場合は、ブラウザ自身が警告用のダイアログボックスを出し、利用者に注意をうながします。

### Sample

```
<script type="text/javascript">
<!--
function LC(go){
        if (go.url.value != "") { location.href=go.url.value }
            else { alert("URLを入力してください。") }
               }
//-->
</script>
  ～中略～
<form name="URL">
<input type="text" name="url" value="http://" size=40>
<input type="button" name= "CF" value=" Go!! " onClick="LC(this.fo
rm)">
</form>
```


<input type="button">：「Formオブジェクト」の「ボタンをリンクに使う」(P.426)

# ページのロードが完了してから次のページをロードする

## location.href ［プロパティ］

Before




After




ページのロードが終ってから10秒後に関数「NEXT1()」が発生し、「location.href」で設定しているURLをブラウザにロードします。
次のページをロードするタイミングは、「setTimeout('NEXT1()',10000)」内の「10000」を変更することで変えられます。
HTMLの<meta>でも自動的にページをロードすることが可能ですが、その場合には回線状態によって表示が遅くなり、ページが完全にロードされる前に次のページがロードされてしまうことがあります。このサンプルの場合は、ページが完全に読み込まれるまでイベントが発生しませんので、そのような問題は回避できます。

### Sample

```
<script type="text/javascript">
<!--
function NEXT1(){ location.href = "http://www.shuwasystem.co.jp/" }
//-->
</script>
  ～中略～
<body onLoad="setTimeout('NEXT1()',10000)">
*ページのロードが完了してから次のページをロードする<p>
ページのロードが終ってから10秒後に別のページへ移ります...。<p>
<img src="robot.jpg" alt="robot.jpg" widht="474" height="198">
</body>
```

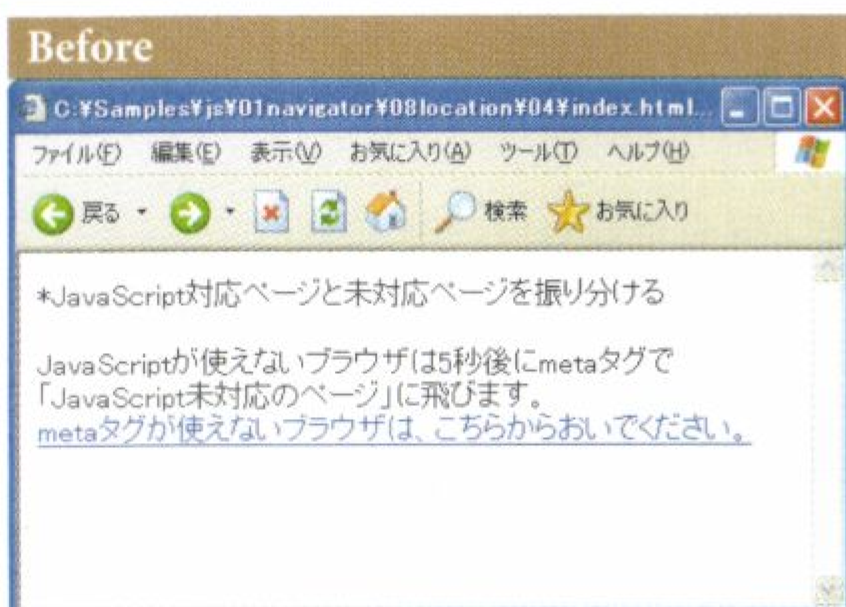

setTimeout()：「windowオブジェクト」の「ステータス行に文字を流す」(P.364)

# JavaScript対応ページと未対応ページを振り分ける

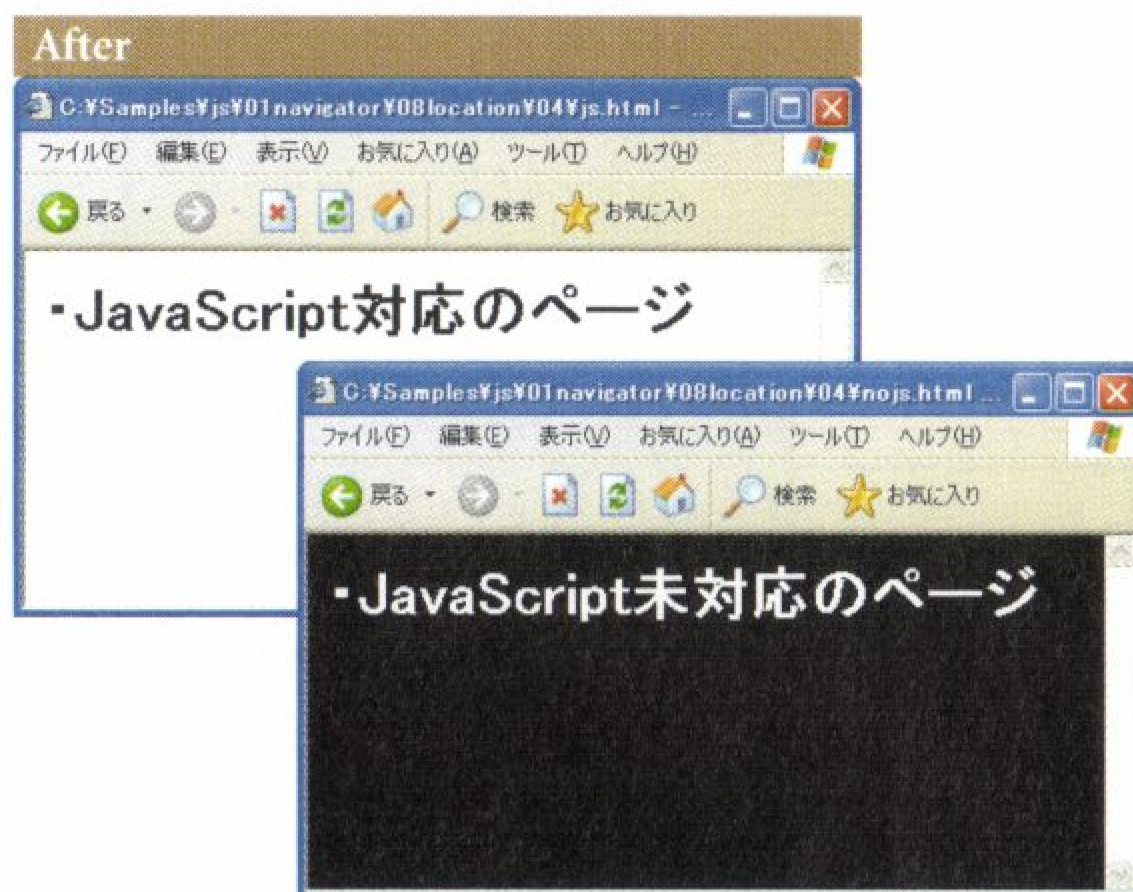
## location.href [プロパティ]



Before




After




サンプルでは、ページのロード時に、JavaScriptに反応するブラウザは「location.href」を理解して「js.html」のページを、JavaScriptに反応しないブラウザはHTMLの<meta>でJavaScript未対応用のページをそれぞれロードし、JavaScriptにも<meta>にも反応しないブラウザには、リンクを設定してページの振り分けを行っています。
実際に試す場合には、この他にも「js.html」「nojs.html」のふたつのHTMLファイルを用意してください。

### Sample

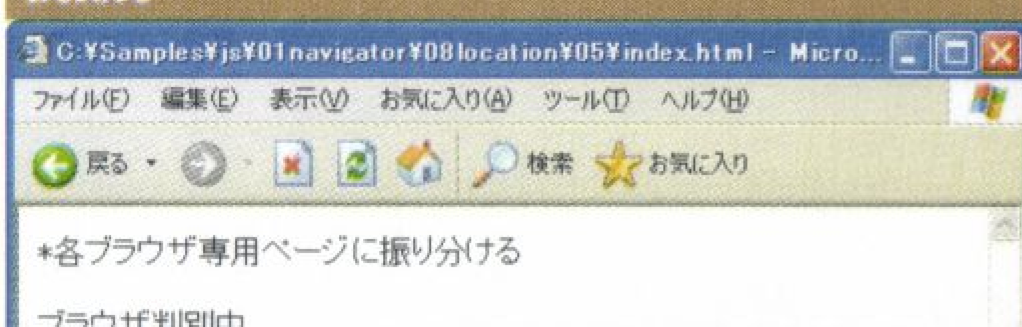
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="refresh"content="5;url=nojs.html">
  ～中略～
<script type="text/javascript">
<!--
function gojs(){ location.href = "js.html" }
//-->
</script>
 ～中略～
<script type="text/javascript">
<!--
gojs()
//-->
</script>
</body></html>
```


<meta>:「基本的な内容」の「自動的にページを読み込む」(P.25)

# 各ブラウザ専用ページに振り分ける

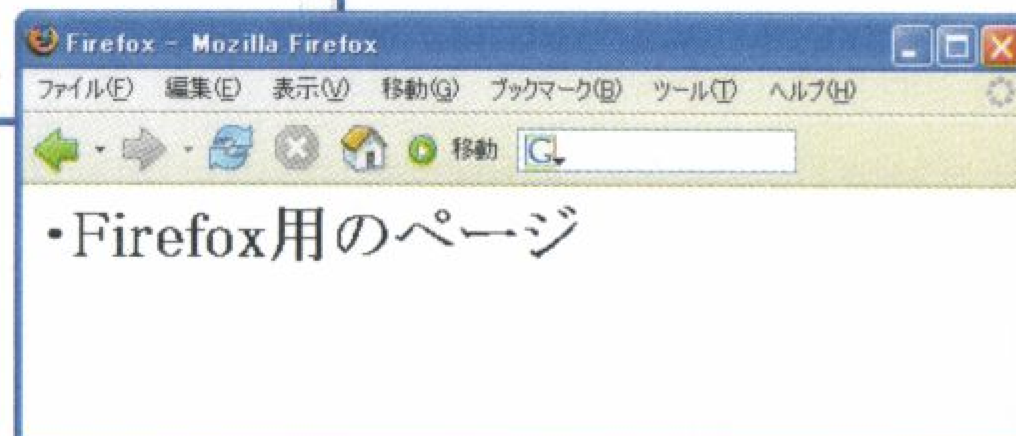
## location.href ［プロパティ］

Before




After




サンプルでは、まずブラウザ名の1番始めの文字を検索して、それを元にNetscape Navigator、Safari、Firefox、Mozillaと、Internet Explorer、Operaを振り分る処理を行っています。そして次に、バージョンの1番始めの文字やユーザーエージェント内に含まれる文字を検索し、Netscape Navigator、Internet Explorerの各バージョンと、Mozilla、Safari、Firefox、Operaを判断し、hrefプロパティを使って、それぞれのブラウザ専用ページへの振り分けを行っています。

Internet Explorer 3.Xは、バージョンとして「2.0」を返すものがあるので、その点も考慮しています。また、Internet Explorer 4.X以外にも、5.Xや6.X、Operaは、バージョンとして「4.0」を返すので、バージョン情報の中から「MSIE 5」「MSIE 6」、ユーザエージェント情報から「Opera」という文字列を検索して振り分けています。

Netscape 6.Xや7.X、Mozilla、Safari、Firefoxは、現在バージョンとして「5.0」を返すので、ユーザエージェント内に含まれる文字列を検索して判断を行っています。

### Sample

```
<script type="text/javascript">
<!--
function GoPege() {
    if( navigator.appName.charAt(0)=="N" ){
        if(navigator.appVersion.charAt(0)==2){ location.href = "NN2
.html" }
        if(navigator.appVersion.charAt(0)==3){ location.href = "NN3
.html" }
        if(navigator.appVersion.charAt(0)==4){ location.href = "NN4
.html" }
```

```
        if(navigator.appVersion.charAt(0)==5){
            if (navigator.userAgent.indexOf("Netscape6/") != -1){ l
ocation.href = "NN6.html" }
            else {
                if (navigator.userAgent.indexOf("Netscape/7") != -1
){ location.href = "NN7.html" }
                else {
                    if (navigator.userAgent.indexOf("Safari") != -1
){ location.href = "Safari.html" }
                    else {
                        if (navigator.userAgent.indexOf("Firefox")
!= -1){ location.href = "Firefox.html" }
                        else { location.href = "Mozilla.html" }
                    }
                }
            }
        }
    }
    if( navigator.appName.charAt(0)=="M" ){
       if(navigator.appVersion.charAt(0)==2){ location.href = "IE3.
html" }
       if(navigator.appVersion.charAt(0)==3){ location.href = "IE3.
html" }
       if(navigator.appVersion.charAt(0)==4){
           if (navigator.userAgent.indexOf("Opera") != -1){ locati
on.href = "Opera.html" }
           else {
               if (navigator.appVersion.indexOf("MSIE 6") != -1){
location.href = "IE6.html" }
               else {
                   if (navigator.appVersion.indexOf("MSIE 5") !=
-1){ location.href = "IE5.html" }
                   else { location.href = "IE4.html" }
               }
           }
       }
    }
}
//-->
</script>
  ～中略～
<script type="text/javascript">
<!--
GoPege()
//-->
</script>
```

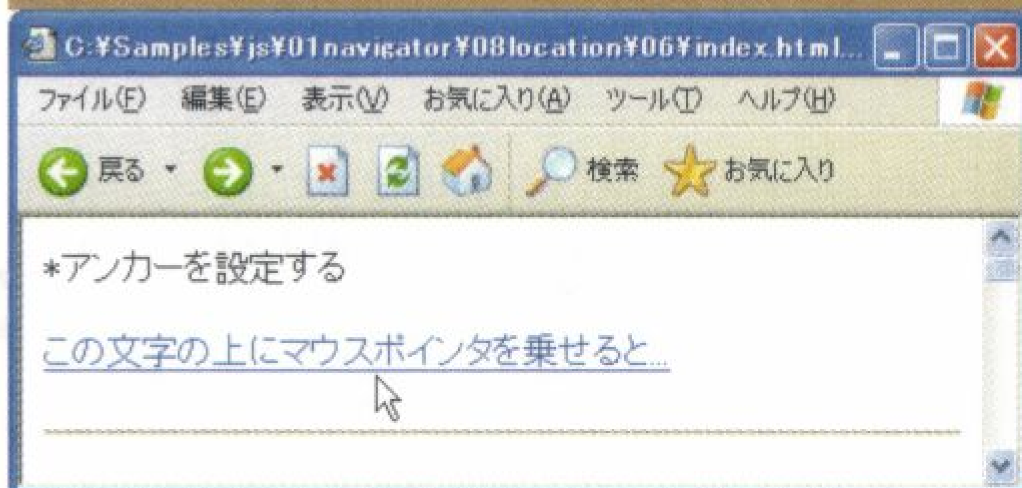

navigator.appName：「navigatorオブジェクト」の「ブラウザ名を取得する」(P.310)
navigator.appVersion：「navigatorオブジェクト」の「ブラウザのバージョンを取得する」(P.311)
navigator.user Agent:「navigatorオブジェクト」の「ブラウザのユーザーエージェントを取得する」(P.312)
charAt()：「stringオブジェクト」の「n番目の文字を抜き出す」(P.555)
indexOf()：「stringオブジェクト」の「先頭から文字列を検索する」(P.558)

IE6.0
IE5.5
IE5.0
IE4.0
N4.06
N4.X
Safari
IE5-mac

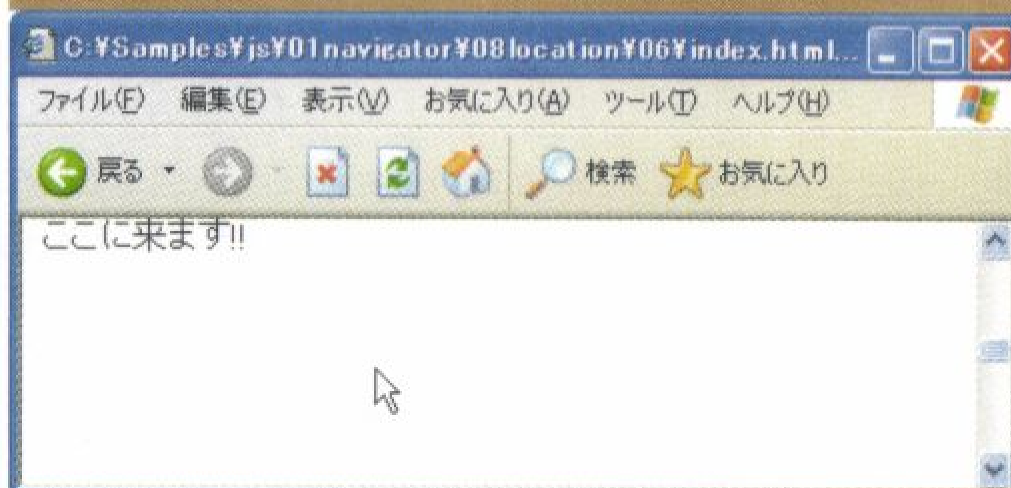
# アンカーを設定する

**onMouseOver="*スクリプト | 関数*"** [イベントハンドラ]
**location.hash** [プロパティ]

Before




After




サンプルでは、リンクの上にマウスポインタが乗った時、関数「LinkMo4('#Go')」が発生して「location.hash」に「#Go」の値が渡され、アンカーで指定された場所にジャンプします。Windows版のブラウザでは、スクロールバーが出ている範囲でしかページが動きません。サンプルでは、移動する範囲を広げるため、ダミーとして<br>タグを入れています。また、Netscape 6.X以降では、このスクリプトは正常に動作しません。

## Sample

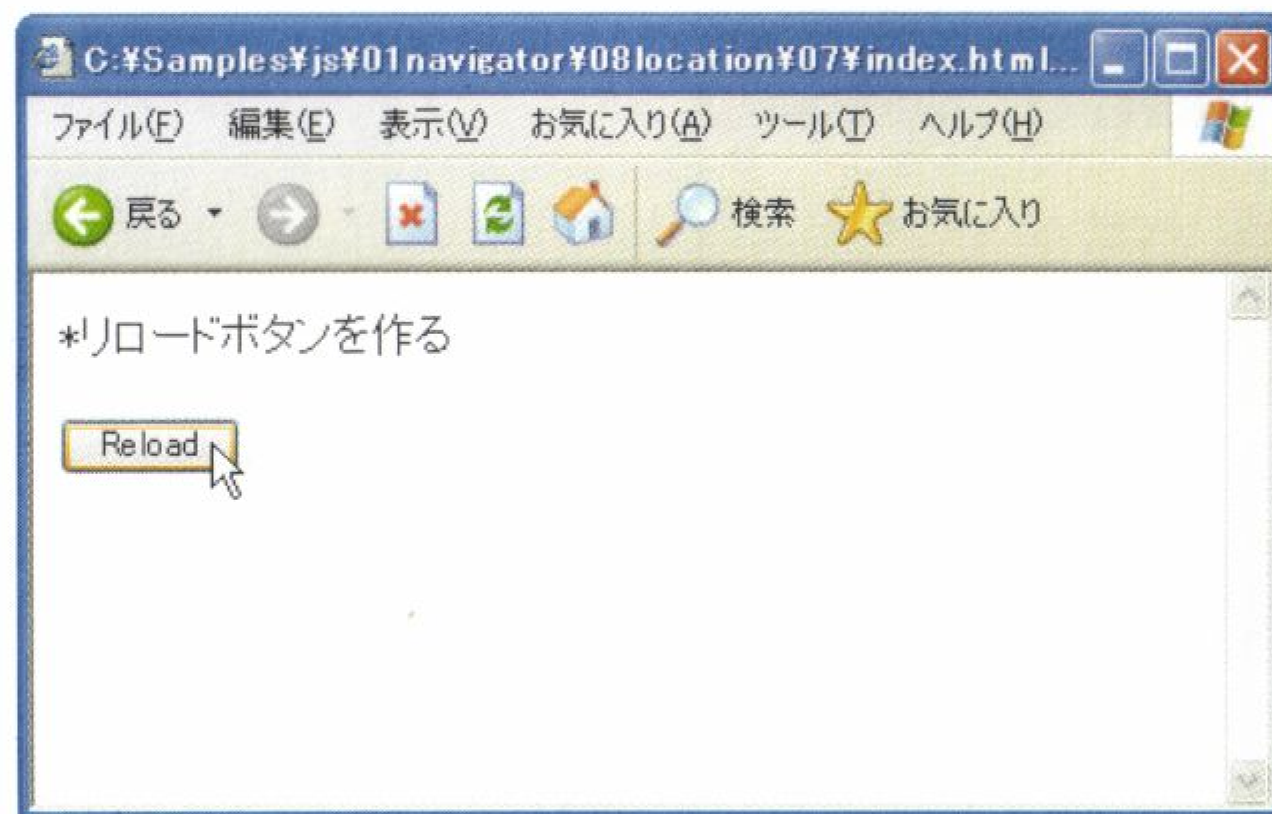
```
<script type="text/javascript">
<!--
function LinkMo4(go) { location.hash = go }
//-->
</script>
 ～中略～
<body>
*アンカーを設定する<P>
<a href="06loc.html" onMouseOver="LinkMo4('#Go')">この文字の上にマウスポインタを乗せると...</a>
<hr>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<hr>
<a name="#Go">ここに来ます!!</a>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
</body>
```

URL情報を利用する

# リロードボタンを作る

## location.reload() [メソッド]





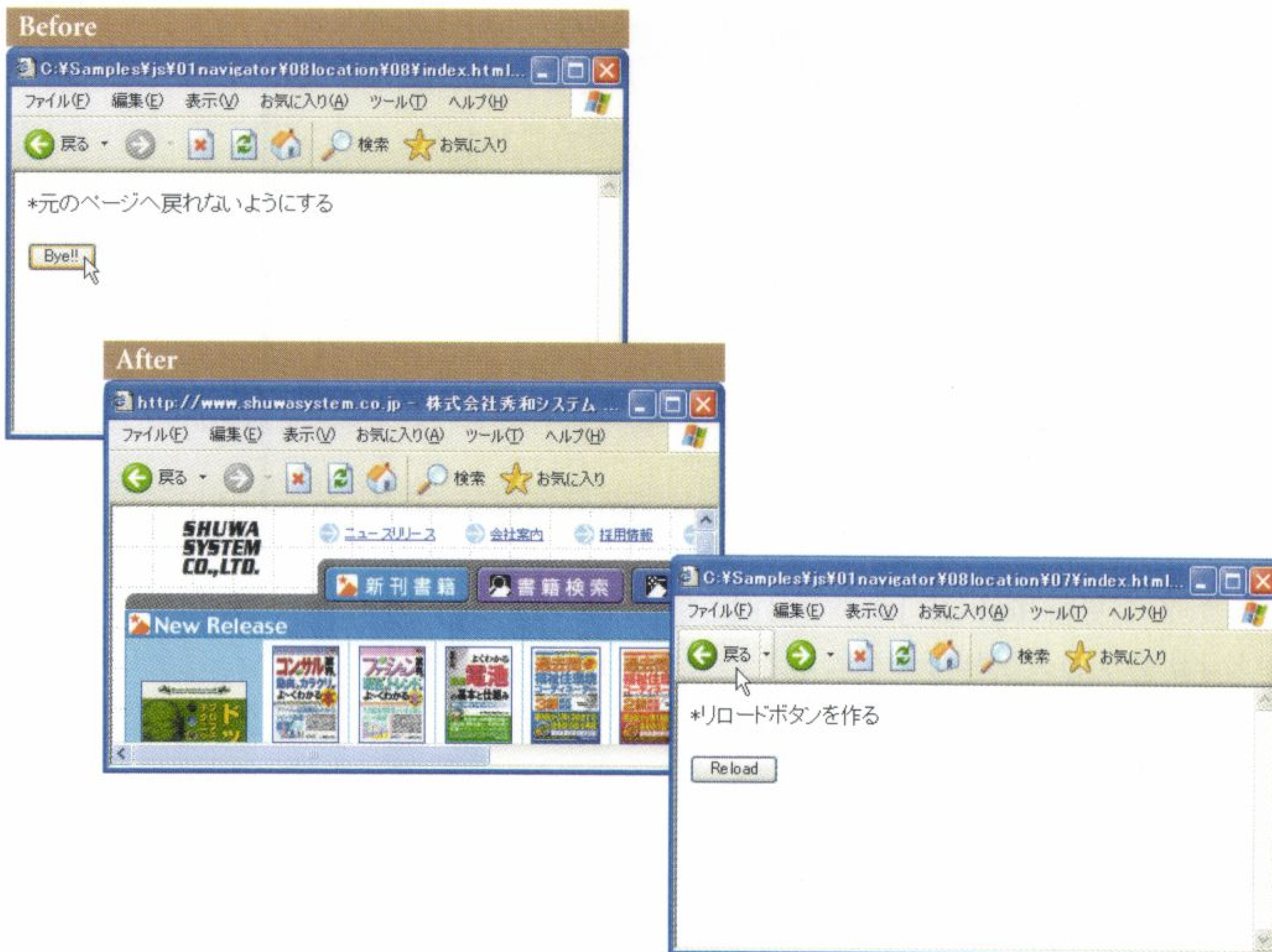

ブラウザの[更新(Reload)]ボタンと同じ働きをするスクリプトです。
ボタンがクリックされたタイミングで「reload()」メソッドが呼ばれ、ページをリロードします。
JavaScript1.1で追加されたメソッドです。

### Sample

```
<form>
<input type="button" value=" Reload " onClick="location.reload()">
</form>
```

# 元のページへ戻れないようにする

## location.replace() [メソッド]




「replace()」メソッドは、現在表示されているURLを「()」内で指定したURLに置き換えます。
したがって、元のページのURLが来歴上に残りませんので、[戻る(Back)]ボタンを使って元のページへ戻って来ることができなくなります。
JavaScript1.1で追加されたメソッドです。

### Sample

```
<form>
<input type="button" value=" Bye!! " onClick="window.location.replace('http://www.shuwasystem.co.jp/')">
</form>
```

JavaScript

# リンクのURLの情報を表示する

| | |
|---|---|
| document.links[インデックス].href | [プロパティ] |
| document.links[インデックス].protocol | [プロパティ] |
| document.links[インデックス].hostname | [プロパティ] |
| document.links[インデックス].pathname | [プロパティ] |
| document.links[インデックス].port | [プロパティ] |
| document.links[インデックス].host | [プロパティ] |
| document.links[インデックス].search | [プロパティ] |
| document.links[インデックス].target | [プロパティ] |

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac




Linkオブジェクトは、ページ上のリンクの0から始まる配列を作成し、その中には各リンクの情報が格納されています。
サンプルでは、リンクがひとつしかないので、そのリンクは「document.links[0]」で参照できます。もしも複数のリンクが存在する場合は、インデックスの部分が上から[0]・[1]・[2]…となります。
「href」プロパティはURL全体の値を、「protocol」プロパティはURL内のhttpやftpなどのプロトコル部分の値を、「hostname」プロパティはURL内のホスト名部分の値を、「pathname」プロパティはURL内のパス名部分の値を、「port」プロパティはURL内の:8080などのポート番号の値を、「host」プロパティはホスト名とポート番号部分の値を、「search」プロパティは?で始まる問い合わせ文字列を、「target」プロパティはtarget属性を、それぞれ持っています。
Linkオブジェクトには、これ以外にも「hash」プロパティ(アンカー)があります。
これらのプロパティは読み出し専用です。

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
＊リンクのURLの情報を表示する
<p>
<a href="http://www.shuwasystem.co.jp/index.html" target="_Open
">秀和システムのホームページ...</a>
</p>
<p>

<script type="text/javascript">
<!--
document.write("URL:",document.links[0].href);
document.write("<br>");
document.write("プロトコル:",document.links[0].protocol);
document.write("<br>");
document.write("ホスト名:",document.links[0].hostname);
document.write("<br>");
document.write("パス名:",document.links[0].pathname);
document.write("<br>");
document.write("コミュニケーションポート番号:",document.links[0].port);
document.write("<br>");
document.write("ホスト名·ポート番号:",document.links[0].host);
document.write("<br>");
document.write("?以降の文字:",document.links[0].search);
document.write("<br>");
document.write("ターゲットウィンドウ:",document.links[0].target);
//-->
</script>

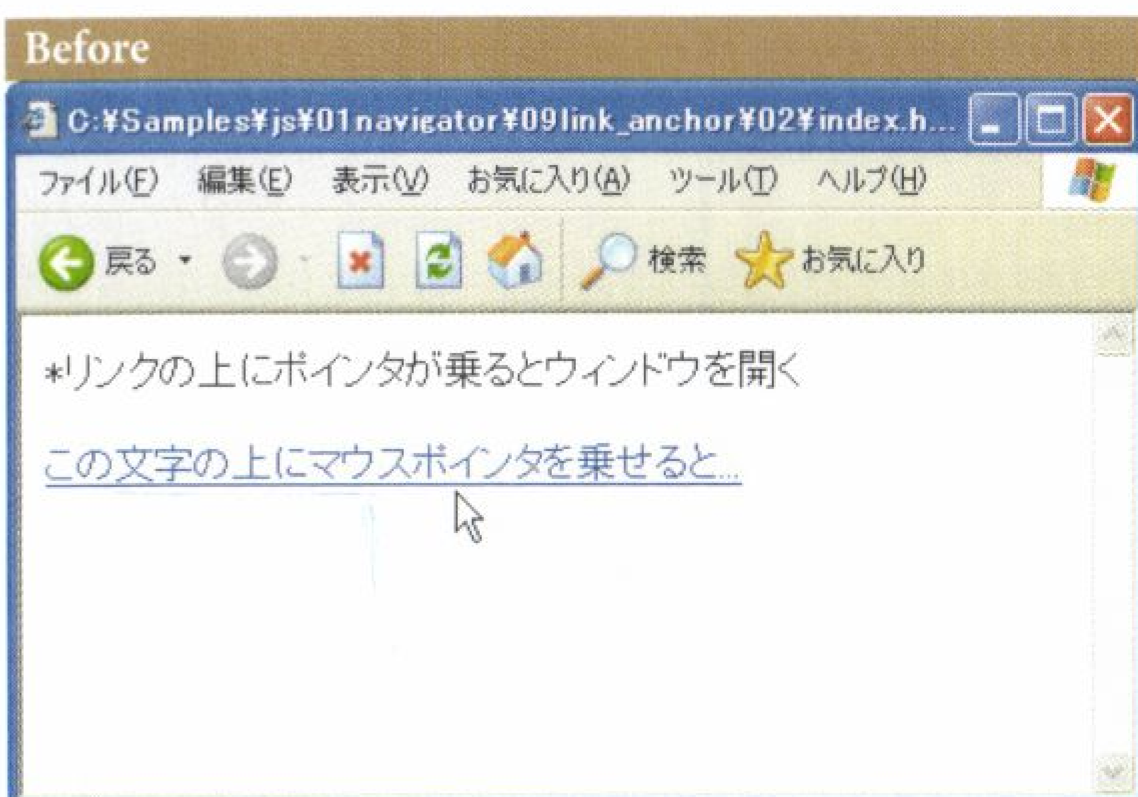
</p>
</body>
</html>
```

# リンクの上にポインタが乗るとウィンドウを開く

**onMouseOver="スクリプト | 関数"** [イベントハンドラ]



Before




After




JavaScriptでは、リンクの中にイベントハンドラを組み込むことができます。
サンプルでは、リンクの中にイベントハンドラ「onMouseOver」を組み込み、マウスカーソルがリンクの上に乗った時に、新しいウィンドウが開くようにしています。

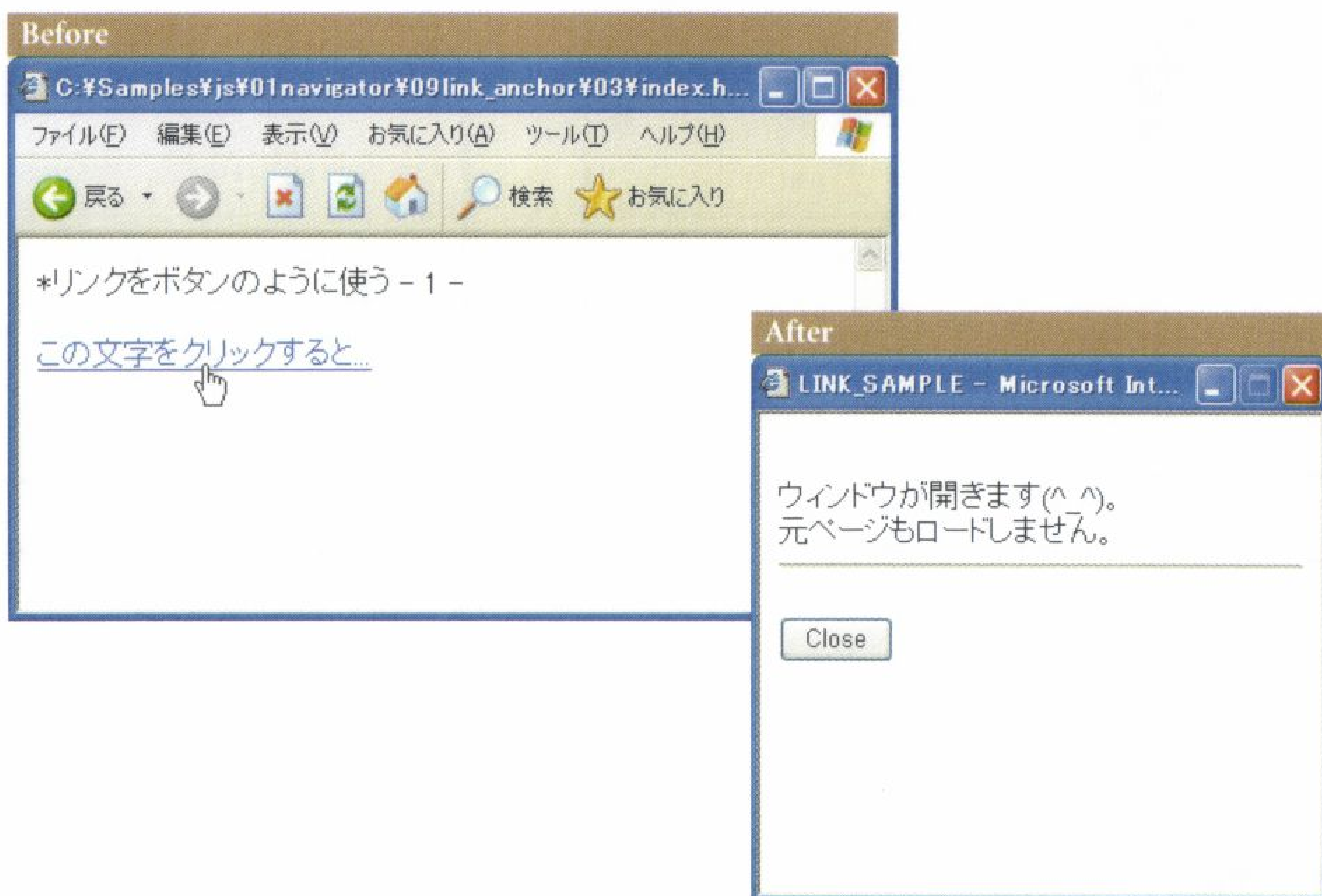
## Sample

```
<script type="text/javascript">
<!--
function LinkMo1(){
    var LM1;
    LM1=window.open("","DocWin1","toolbar=no,location=no,directori
es=no,width=300,height=250");
    LM1.document.write("<html><head><title>LINK_SAMPLE</title>");
    LM1.document.write("</head>");
    LM1.document.write("<body>");
    LM1.document.write("<br>");
    LM1.document.write("ウィンドウが開きます(^_^)。");
    LM1.document.write("<hr>");
    LM1.document.write("<form>");
    LM1.document.write("<input type='button' name= 'Wcl1' value='
Close ' onClick='window.close()'>");
    LM1.document.write("</form>");
    LM1.document.write("</body>");
    LM1.document.write("</html>");
    LM1.document.close();
}
//-->
</script>
  ～中略～
<p><a href="#" onMouseOver="LinkMo1()">この文字の上にマウスポインタを乗せる
と...</a></p>
```

# リンクをボタンのように使う - 1 -

## href="javascript:関数"






リンクのURLを指定する部分で「javascript:関数」と指定すると、リンクをクリックしたタイミングで関数を発生させることができます。
サンプルでは、リンクをクリックすると関数「LinkMo2()」が発生して、元のページはそのままで新しいウィンドウが開きます。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function LinkMo2(){
    var LM1;

    LM1=window.open("","DocWin2","toolbar=no,location=no,directori
es=no,width=300,height=250");
    LM1.document.write("<html><head><title>LINK_SAMPLE</title>");
    LM1.document.write("</head>");
```

```
    LM1.document.write("<body>");
    LM1.document.write("<br>");
    LM1.document.write("ウィンドウが開きます(^_^)。");
    LM1.document.write("<br>");
    LM1.document.write("元ページもロードしません。");
    LM1.document.write("<hr>");
    LM1.document.write("<form>");
    LM1.document.write("<input type='button' name= 'Wcl2' value='
Close ' onClick='window.close()'>");
    LM1.document.write("</form>");
    LM1.document.write("</body>");
    LM1.document.write("</html>");
    LM1.document.close();
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

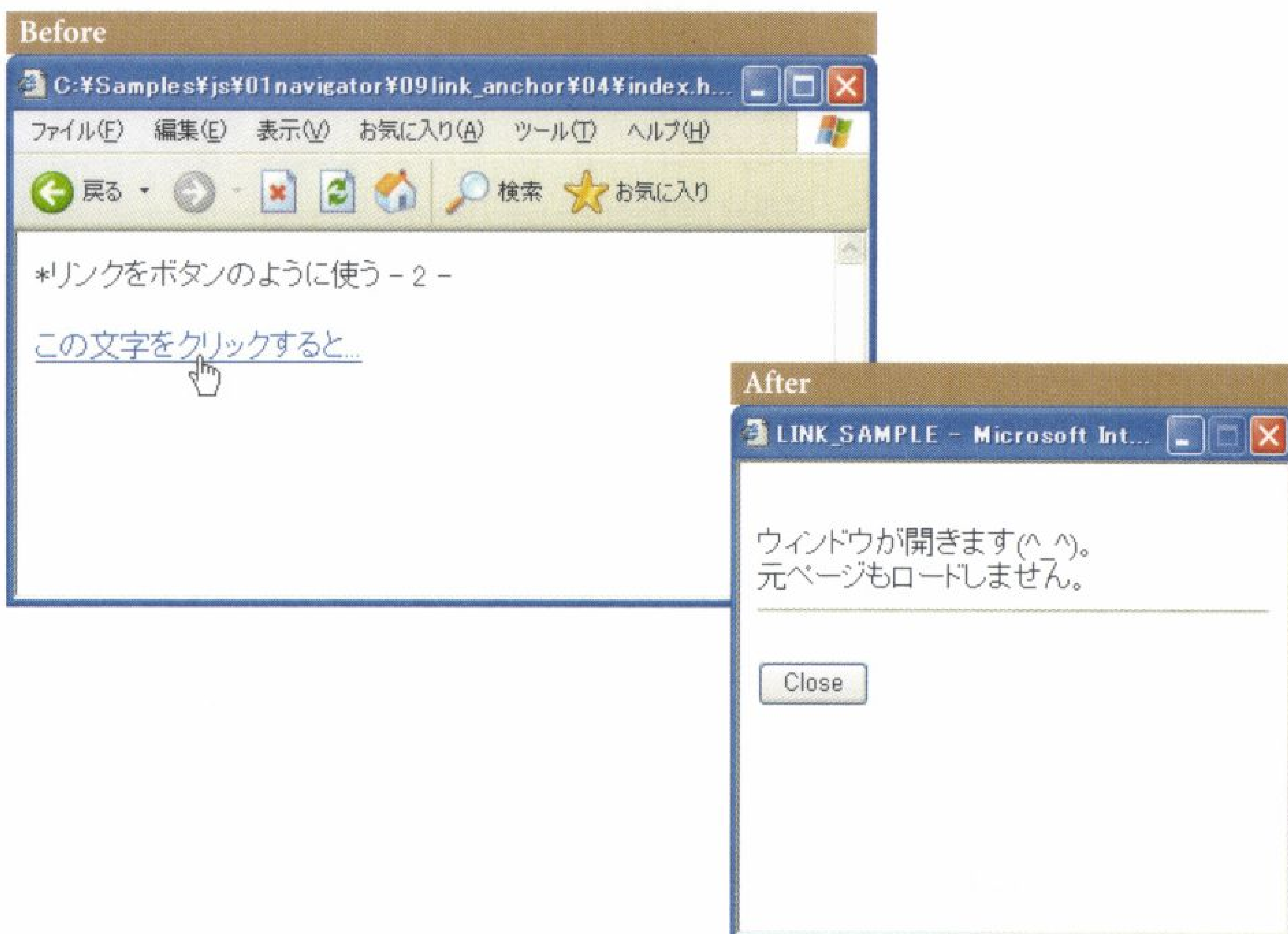
</head>
<body>
*リンクをボタンのように使う - 1 -
<p>
<a href="javascript:LinkMo2()">この文字をクリックすると...</a>
</p>
</body>
</html>
```

JavaScript

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# リンクをボタンのように使う - 2 -

**onClick="*スクリプト | 関数*"** [イベントハンドラ]
**return false**




リンク上で「onClick」を使用した場合、リンクをクリックした時に関数が発生するのと同時に、リンクの動作も発生してしまいます。しかし、JavaScript1.1からは、「return false」を返すと、その時点で関数の処理とリンクの動作を中止できるようになりました。サンプルでは、リンクがクリックされると関数「LinkMo4()」が評価されて、ウィンドウを開く処理を行います。その後、「return false」でイベントの処理を中止するのと同時に、リンクの動作も中止するようにしているので、他のリンクへ飛ぶ動作は発生しません。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function LinkMo3(){
    var LM1;
    LM1=window.open("","DocWin3","toolbar=no,location=no,directori
es=no,width=300,height=250");
```

リンク・アンカー情報を利用する

```
    LM1.document.write("</head><title>LINK_SAMPLE</title>");
    LM1.document.write("</head>");
    LM1.document.write("<body>");
    LM1.document.write("<br>");
    LM1.document.write("ウィンドウが開きます(^_^)。");
    LM1.document.write("<br>");
    LM1.document.write("元ページもロードしません。");
    LM1.document.write("<hr>");
    LM1.document.write("<form>");
    LM1.document.write("<input type='button' name= 'Wcl3' value='
Close ' onClick='window.close()'>");
    LM1.document.write("</form>");
    LM1.document.write("</body>");
    LM1.document.write("</html>");
    LM1.document.close();
    return false;
}
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*リンクをボタンのように使う - 2 -
<p>
<a href="#" onClick="return LinkMo3()">この文字をクリックすると...</a>
</p>
</body>
</html>
```

重要

## リンクでイベントハンドラを使用する時の注意

リンク上で「onClick」などのイベントハンドラを使う時には、必ずリンクの指定をしてください。

リンクの指定がされていない時にリンクがクリックされると、サーバーの設定によっては、そのHTMLファイルが納められているディレクトリの一覧が表示されてしまうような場合もあります。

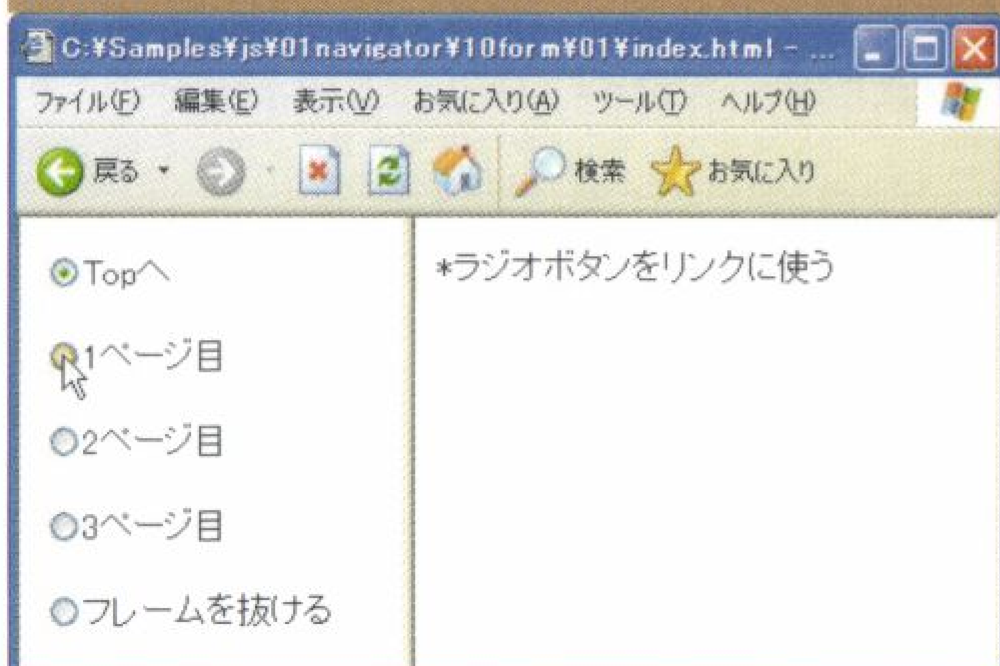
Netscape Navigator 3.0以上では、「return false」を使えば問題は解消されますが、その他のブラウザでは「return false」は無視されます。

もしも、リンクをボタンのように使い、クリックしても他のページへ飛びたくないような時には、<a href="#">といったように、リンクに「#」を設定するとよいでしょう。

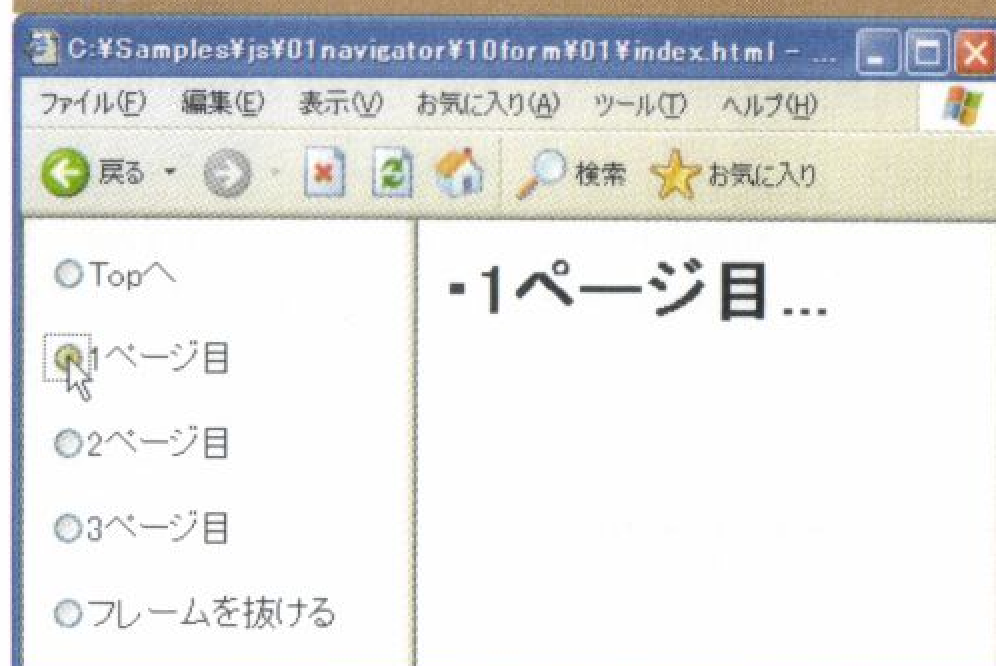
# ラジオボタンをリンクに使う

**<input type="radio" name= "radioオブジェクト名" value="値" イベントハンドラ>**

Before

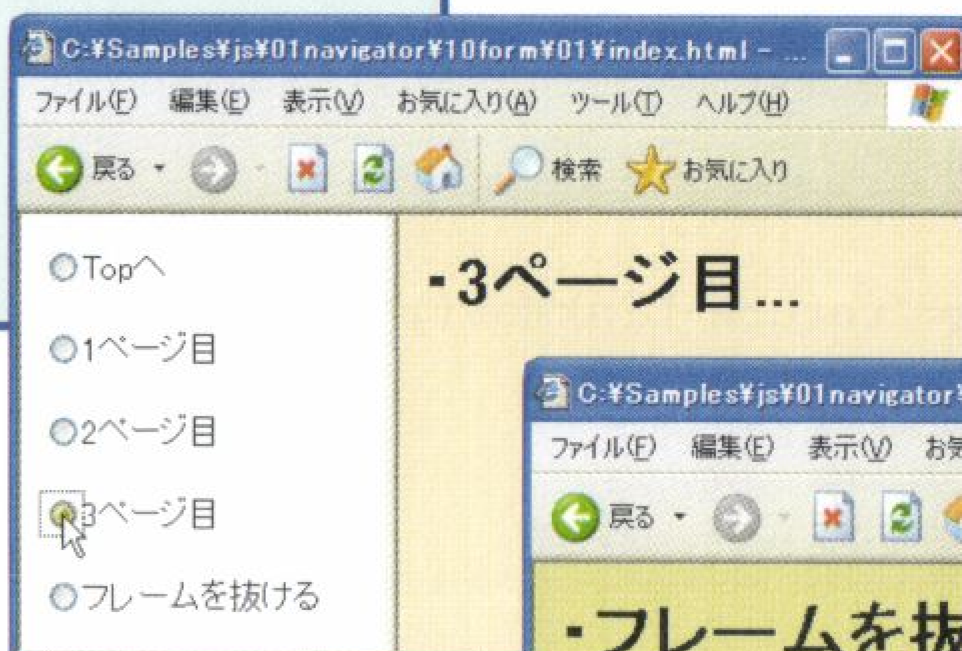


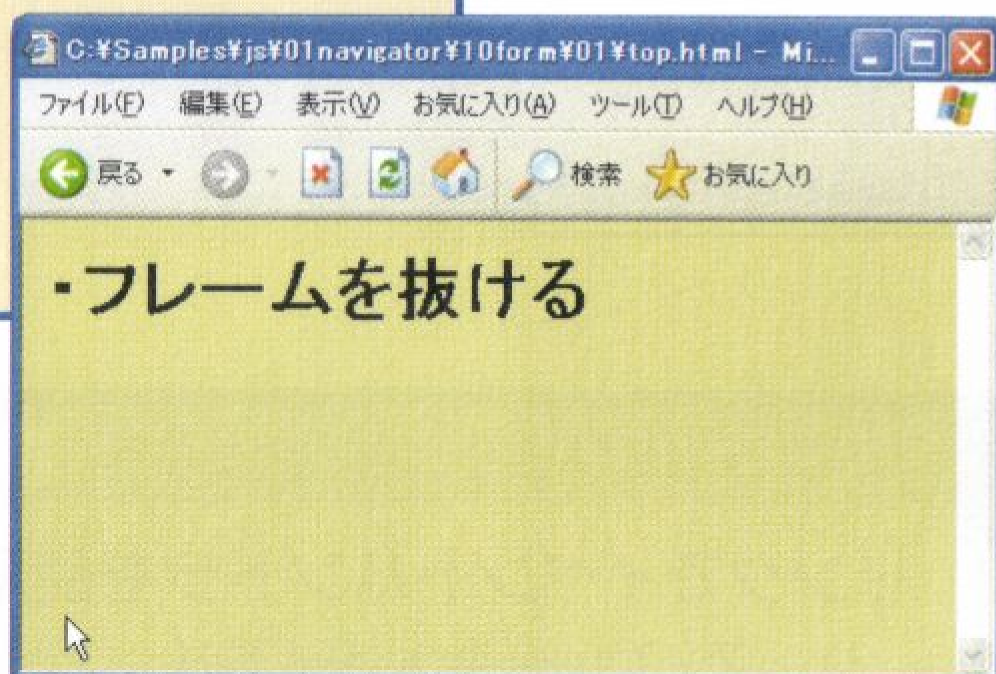

After










サンプルでは、ラジオボタンにイベントハンドラ「onClick」を設定することにより、ラジオボタンがクリックされると関数が発生し、URLが引き渡されて、フレーム名「f2」にページがロードされます。

locationオブジェクトの「href」プロパティを使ってフレームにページを読み込む場合、フレームを抜けてページを表示するには、サンプルのように「parent.top.location.href=URL」と、「top」プロパティを指定します。

実際に試す場合には、この他にも「f2.html」「page1.html」～「page3.html」「top.html」の5つのHTMLファイルを用意してください。

## Sample

**【フレームウィンドウ】**

```
<frameset cols="180,*">
    <frame src="f1.html" name="f1">
    <frame src="f2.html" name="f2">
</frameset>
<noframes>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</noframes>
```

**【f1.html】**

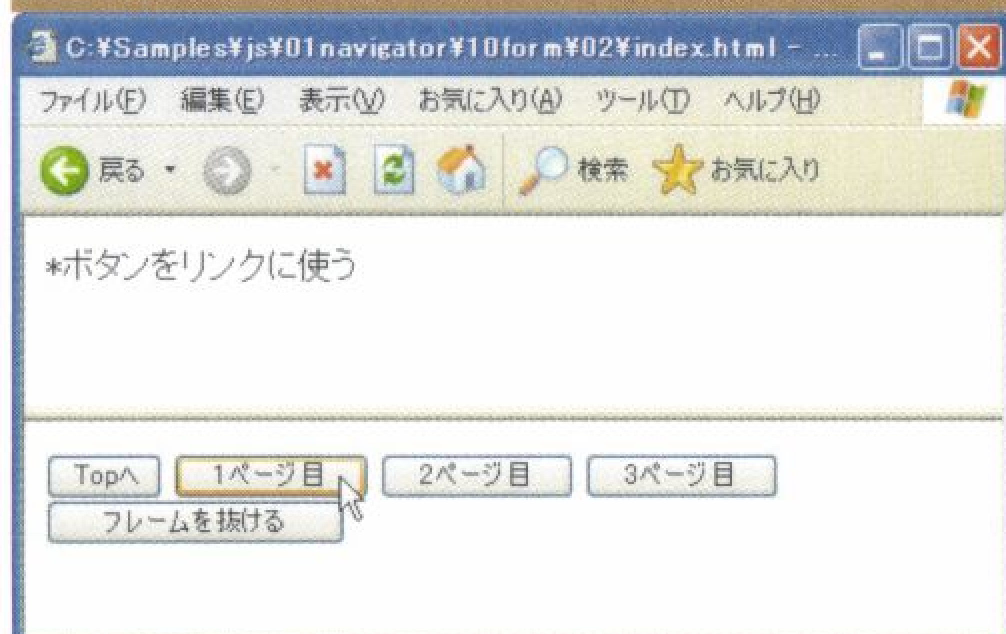
```
<script type="text/javascript">
<!--
function P1(w1) { parent.f2.location.href=w1 }
function TP(w2) { parent.top.location.href=w2 }
//-->
</script>
 ～中略～
</head>
<body>
<form>
<p>
 <input type="radio" name= "FRGo" value="FR" onClick="P1('f2.html'
se)">)" checked>Topへ
</p>
<p>
 <input type="radio" name= "FRGo" value="FR" onClick="P1('page1.ht
se)">ml')">1ページ目
</p>
<p>
 <input type="radio" name= "FRGo" value="FR" onClick="P1('page2.ht
se)">ml')">2ページ目
</p>
<p>
 <input type="radio" name= "FRGo" value="FR" onClick="P1('page3.ht
se)">ml')">3ページ目
</p>
<p>
 <input type="radio" name= "FRGo" value="FR" onClick="TP('top.html
se)">')">フレームを抜ける
</p>
</form></body>
```


<input type="radio">：「フォーム」の「ラジオボタンを作る」(P.124)
parent：「frameオブジェクト」の「入力されたURLを別フレームに表示する」(P.374)
parent.フレーム名.location.href=URL：「frameオブジェクト」の「複数のフレームを同時に変更する - ボタンを使う -」(P.376)

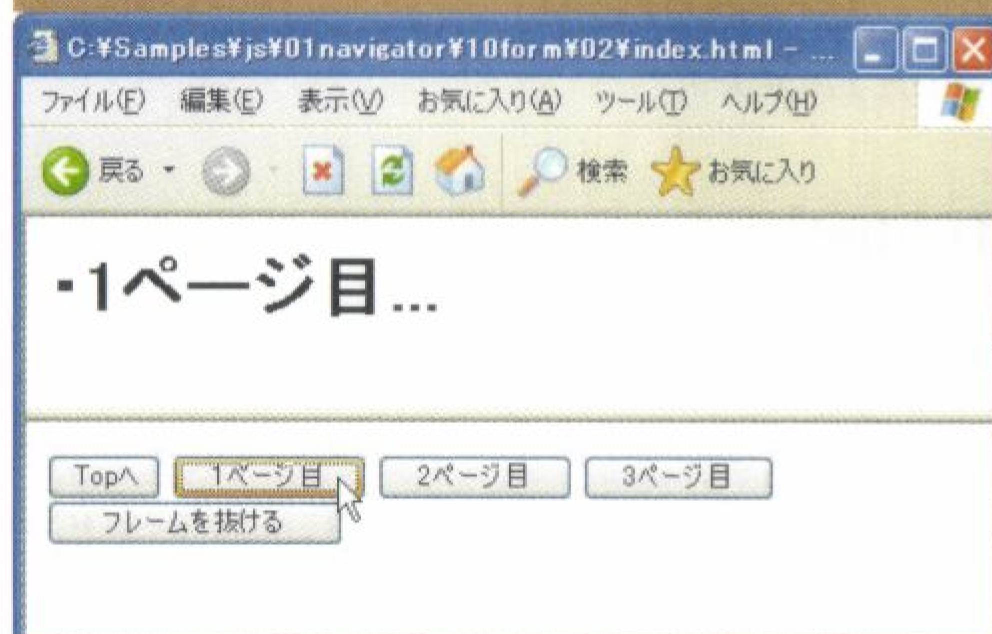
# ボタンをリンクに使う

**<input type="button" name= "*button オブジェクト名*" value="*値*" *イベントハンドラ*>**

Before



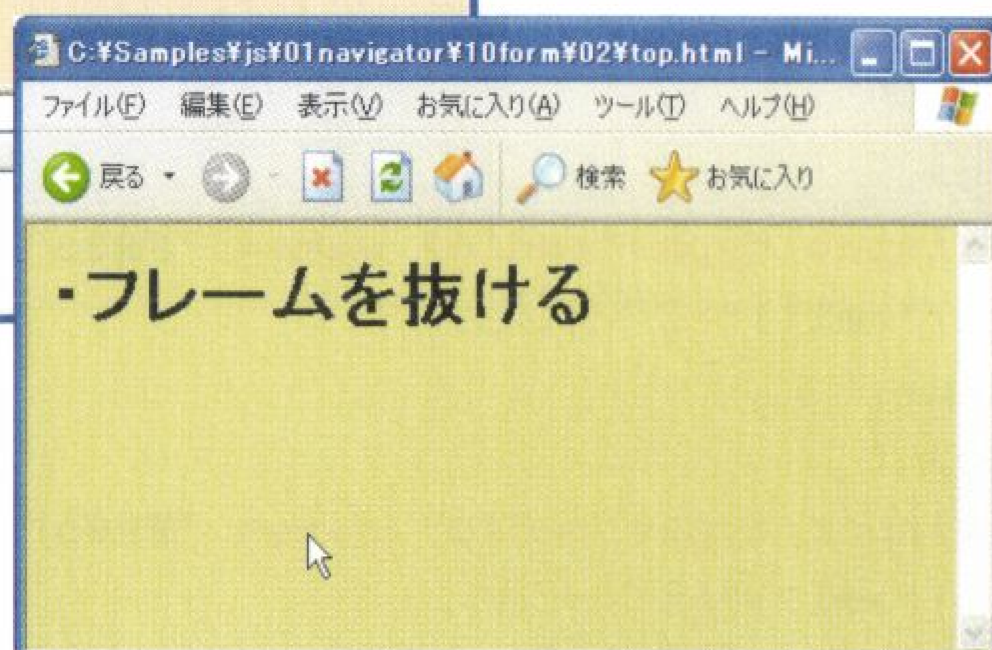

After










サンプルでは、ボタンにイベントハンドラ「onClick」を設定することにより、ボタンがクリックされると関数が発生し、URLが引き渡されて、フレーム名「f1」にページがロードされます。
windowオブジェクトの「open()」メソッドを使ってフレームにページを読み込む場合、フレームを抜けてページを表示するには、サンプルのように「window.open(URL,"_top")」と、ウィンドウ名に「_top」を指定します。
実際に試す場合には、この他にも「f1.html」「page1.html」～「page3.html」「top.html」の5つのHTMLファイルを用意してください。

## Sample

**【フレームウィンドウ】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" >
<html><head>
<title></title>
</head>
<frameset rows="*,100">
    <frame src="f1.html" name="f1">
    <frame src="f2.html" name="f2">
</frameset>
<noframes>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</noframes>
</html>
```

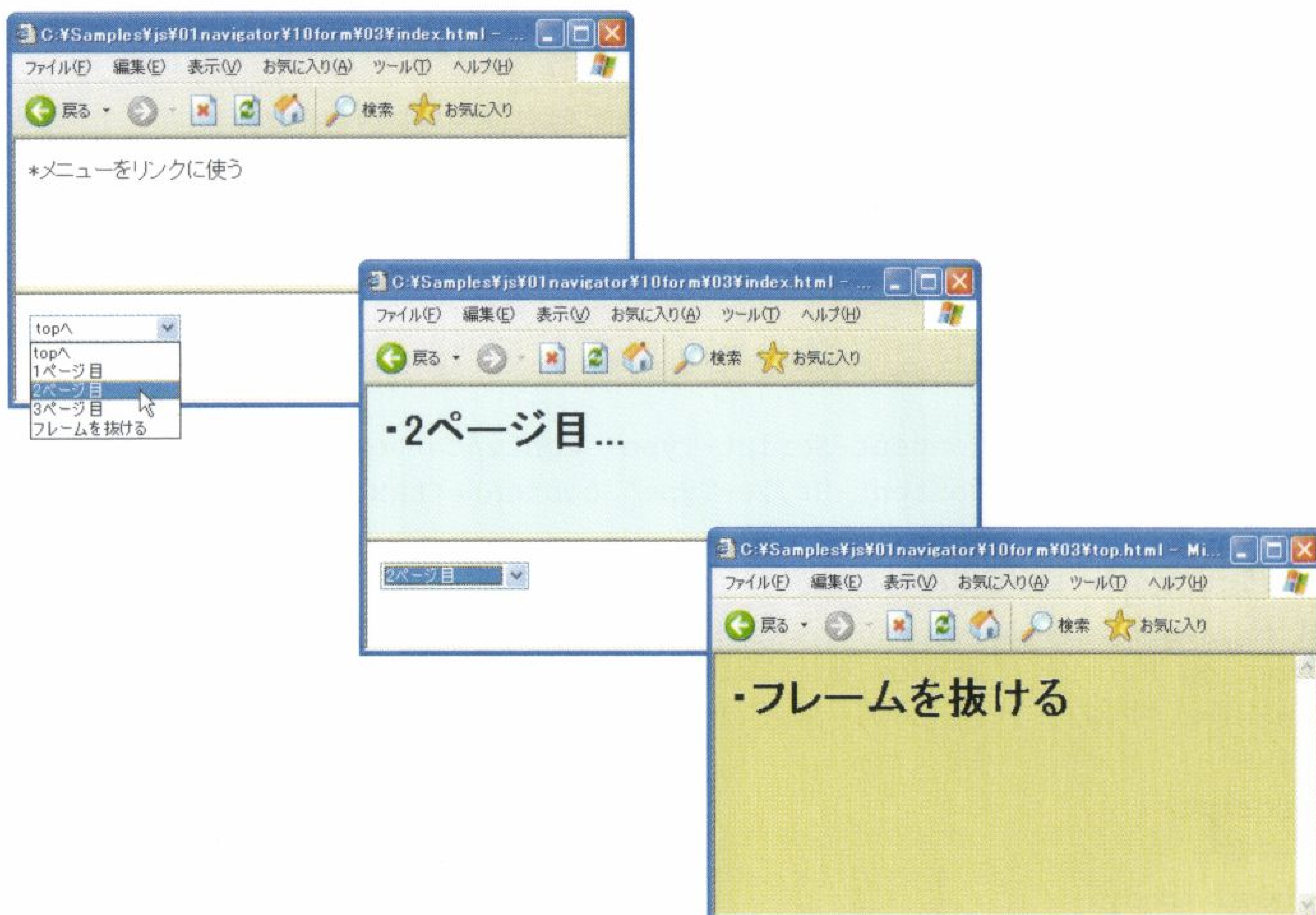
**【f2.html】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function P1(w1) { window.open(w1,"f1") }
function TP(w2) { window.open(w2,"_top") }
//-->
</script>
 ～中略～
</head><body>
<form>
    <input type="button" name= "FBGo1" value=" Topへ" onClick="P1
('f1.html')">
    <input type="button" name= "FBGo2" value=" 1ページ目 " onClick=
"P1('page1.html')">
    <input type="button" name= "FBGo3" value=" 2ページ目 " onClick=
"P1('page2.html')">
    <input type="button" name= "FBGo4" value=" 3ページ目 " onClick=
"P1('page3.html')">
    <input type="button" name= "FBGo4" value=" フレームを抜ける " onCl
ick="TP('top.html')">
</form>
</body></html>
```


<input type="button">：「フォーム」の「汎用的なボタンを作る」(P.117)
parent：「frameオブジェクト」の「入力されたURLを別フレームに表示する」(P.374)
window.open("URL","フレーム名")：「frameオブジェクト」の「複数のフレームを同時に変更する - リンクを使う -」(P.378)

# メニューをリンクに使う

<select name="selectオブジェクト名" イベントハンドラ>
<option> 文字列
フォームオブジェクト名.セレクトオブジェクト名.selectedIndex
onChange="スクリプト | 関数" [イベントハンドラ]




サンプルでは、フォームの内容に変化があった時のイベントを取得して処理を発生するイベントハンドラ「onChange」によって、メニューのオプションが変更された時に、関数「FC()」の処理が発生します。

JavaScriptは、自動的に0から始まるオプションの配列を作成しているので、どのオプションが選ばれたかは、それで参照することができます。具体的には、「WO.FSGo.selectedIndex == 0」は、セレクト名「FSGo」のインデックス1番目、つまり「<option> Topへ」を指します。この値が真(True)の時は、そこで指定しているURL「f1.html」がフレーム「f1」に読み込まれます。

実際に試す場合には、この他にも「f1.html」「page1.html」〜「page3.html」「top.html」の5つのHTMLファイルを用意してください。

## Sample

**【フレームウィンドウ】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" >
<html>
<head>
```

```
<title></title>
</head>
<frameset rows="*,80">
    <frame src="f1.html" name="f1">
    <frame src="f2.html" name="f2">
</frameset>
<noframes>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</noframes>
</html>
```

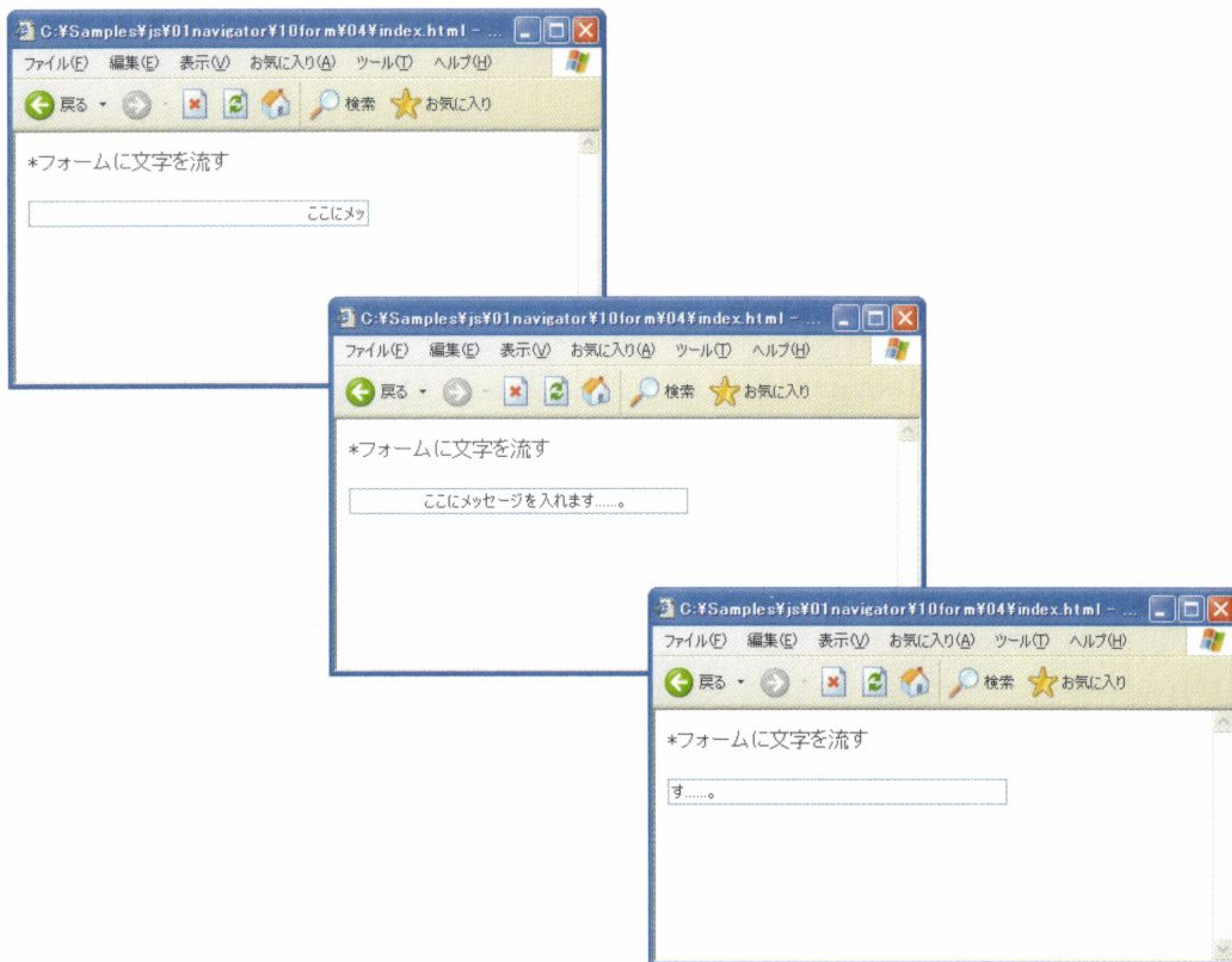
**【f2.html】**

```
<script type="text/javascript">
<!--
function FC(WO) {
    if (WO.FSGo.selectedIndex == 0) { parent.f1.location.href = 
"f1.html" }
    if (WO.FSGo.selectedIndex == 1) { parent.f1.location.href = 
"page1.html" }
    if (WO.FSGo.selectedIndex == 2) { parent.f1.location.href = 
"page2.html" }
    if (WO.FSGo.selectedIndex == 3) { parent.f1.location.href = 
"page3.html" }
    if (WO.FSGo.selectedIndex == 4) { parent.top.location.href = 
"top.html" }
}
//-->
</script>
 ～中略～
</head>
<body>
<form>
  <select name="FSGo" onChange="FC(this.form)">
    <option> topへ
    <option>1ページ目
    <option>2ページ目
    <option>3ページ目
    <option> フレームを抜ける
   </select>
</form>
</body>
```


<select>：「フォーム」の「メニューを作る」(P.126)
parent：「frameオブジェクト」の「入力されたURLを別フレームに表示する」(P.374)
parent.フレーム名.location.href=URL：「frameオブジェクト」の「複数のフレームを同時に変更する - ボタンを使う -」(P.376)

# フォームに文字を流す

<form name="フォームオブジェクト名">
<input type="text" name="textオブジェクト名" size=ピクセル>
document.フォーム名.テキストフォーム名.value　　[プロパティ]




サンプルでは、文字をスクロールさせる部分は、「windowオブジェクト」の「ステータス行に文字を流す」(P.364)と同じです。
ただし、文字をスクロールさせるのを、ステータス行ではなくフォーム内のテキスト欄にするために、「window.status」と指示していた部分が「document.Fmess.fmess.value」となっています。これは、documentオブジェクト内の「Fmess」という名前のFormオブジェクトにある「fmess」という名前のtextオブジェクト名の値であることを表し、そこに文字列を代入しています。
オブジェクトの階層が深くて少々ややこしく感じるかもしれませんが、常にオブジェクトの階層関係を念頭に置いてスクリプトを書くようにすればよいでしょう。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
var TC = 0;
var Fm1 = "
";
var Fm2 = "ここにメッセージを入れます......。";
var Fm = Fm1+Fm2;
function FMess() {
    if (TC < 1000) { //ここの数値を変える事によってスクロールする時間が変わります
        TC++;
                    document.Fmess.fmess.value = Fm;
                    Fm = Fm.substring(2,Fm.length) + Fm.substring
(0,2);
                    setTimeout("FMess()",300);
                   }
   else { document.Fmess.fmess.value = "" }
}
//-->
</script>

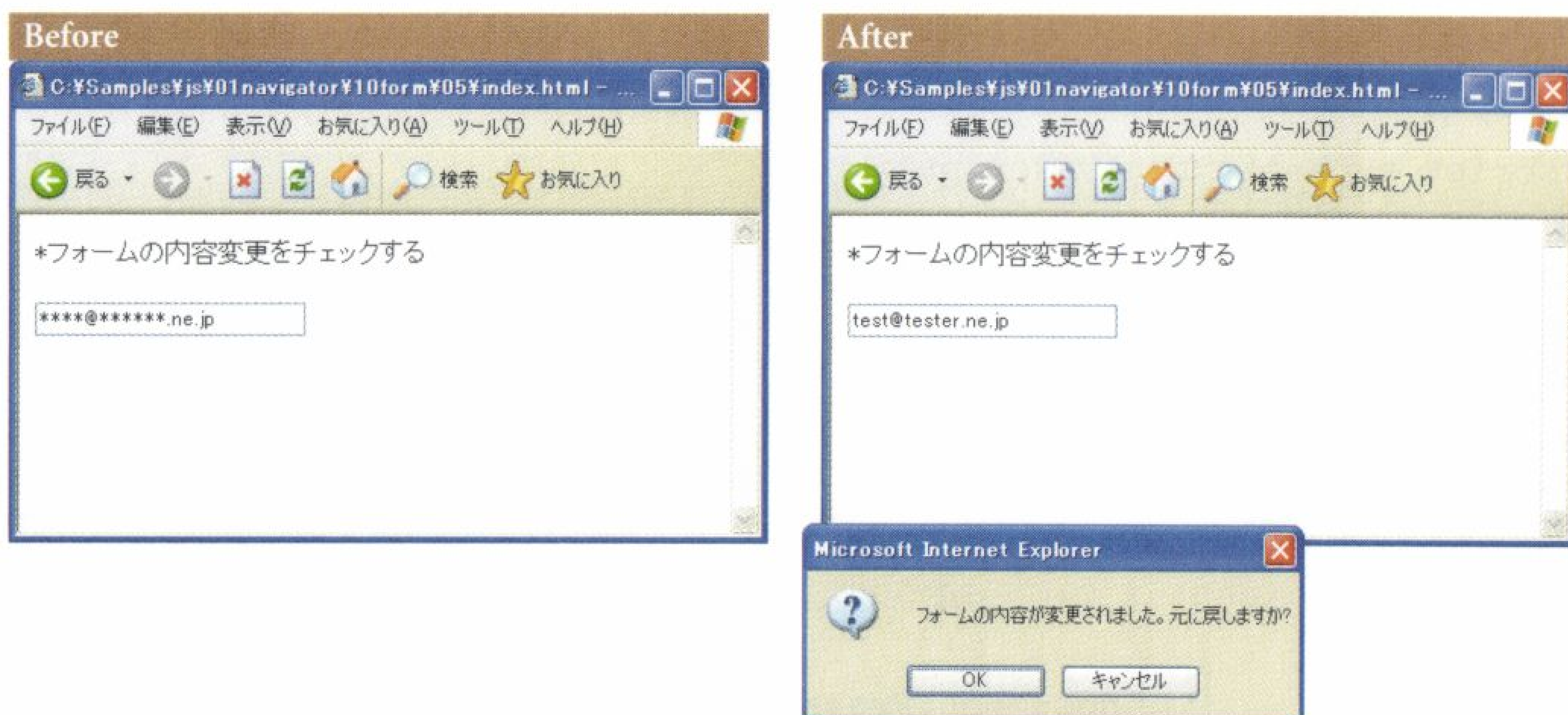
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body onLoad="FMess()">
*フォームに文字を流す
<p>
<form name="Fmess">
<input type="text" name="fmess" SIZE=50>
</form>
</p>
</body>
</html>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# フォームの内容変更をチェックする

**onChange="スクリプト | 関数"** [イベントハンドラ]




サンプルでは、イベントハンドラ「onChange」を使って、フォームの内容に変化があったかどうかをチェックしています。
もしもテキストエリア内の値が変えられた場合は、フォームを抜ける時に「onChange」が関数を発生し、テキストエリアの値を元に戻すかどうかを確認するダイアログボックスを出す処理を実行します。

## Sample

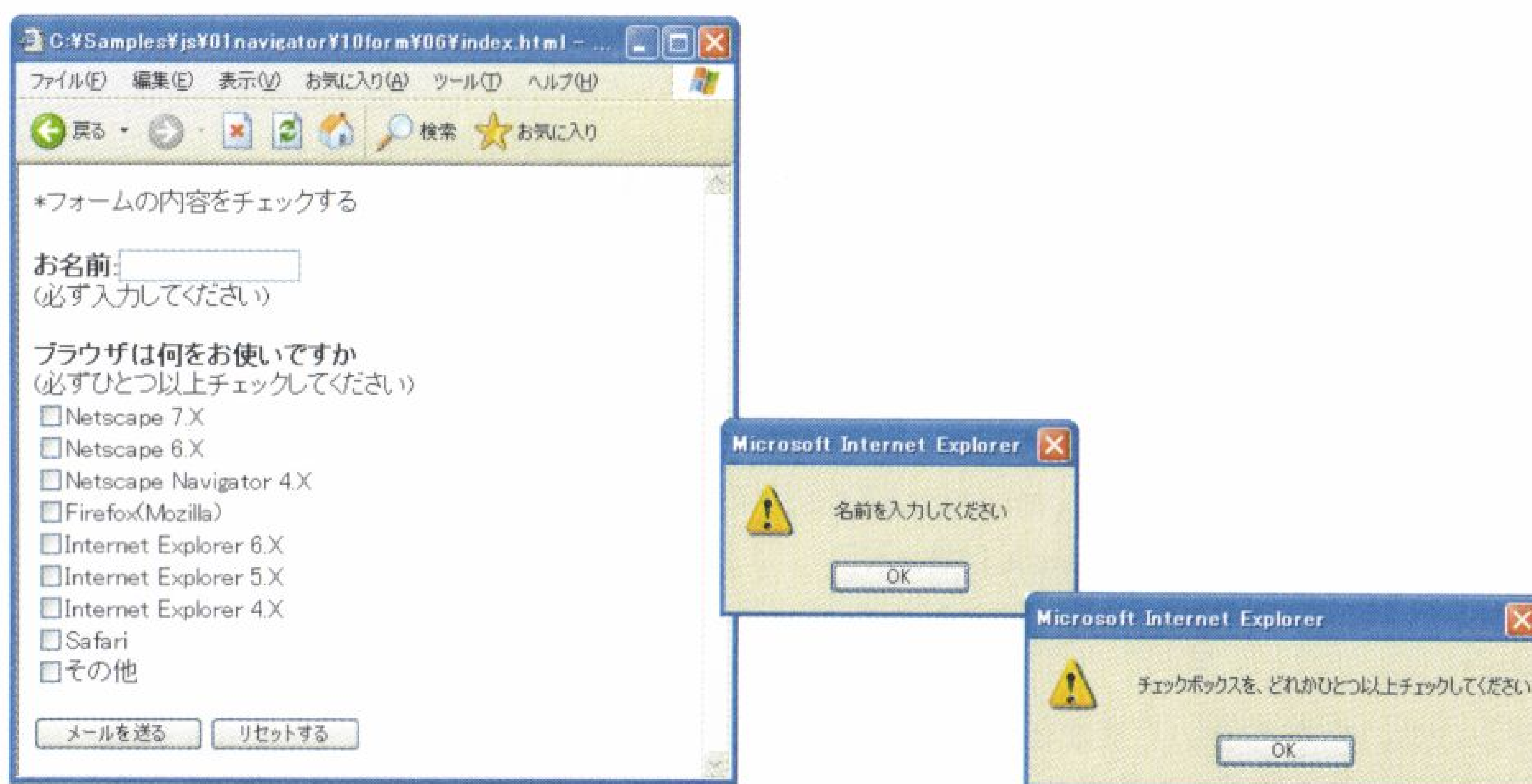
```
<script type="text/javascript">
<!--
function OC() {
    if ( confirm ("フォームの内容が変更されました。元に戻しますか？") ) {
        document.OnC.onc.value = "****@******.ne.jp";
    }
}
//-->
</script>
  ～中略～
<body>
*フォームの内容変更をチェックする
<form name="OnC">
<input type="text" name="onc" value = "****@******.ne.jp" size=30
 onChange="OC()">
</form>
</body>
```

<input type="text">：「フォーム」の「1行の入力フィールドを作る」(P.120)
confirm()：「windowオブジェクト」の「確認ボタン付きのダイアログボックスを開く」(P.334)

# フォームの内容をチェックする

<form name="フォームオブジェクト名" action="送信先" method="post" イベントハンドラ>
<input type="checkbox" name="checkboxオブジェクト名" value="値" イベントハンドラ>
<input type="reset" value="値" イベントハンドラ>






サンプルでは、[メールを送る]ボタンが押された時に、イベントハンドラ「onSubmit」が関数の処理を発生して、フォーム内のチェックを行っています。
もしもその時に、名前を書く欄に何も記入されていなかったり、チェックボックスがひとつもチェックされていない場合は、警告用のダイアログボックスを開きます。
チェックボックスは、1番始めにすべてのチェックボックスに「false(偽)」の値を代入し、チェックボックスがクリックされる度にその反対の値、「false(偽)」だったら「true(真)」、「true(真)」だったら「false(偽)」を代入することにより判定しています。また、[リセットする]ボタンが押された時は、すべてのチェックボックスに「false(偽)」の値が代入されるようにしています。
実際に試す場合には、サンプルの「****@******.ne.jp」の部分を、メールを受け取りたいメールアドレスに変更するか、データを受け取るCGIサーバーを設定してください。

## Sample

```
<script type="text/javascript">
<!--
var f1=false;
var f2=false;
var f3=false;
var f4=false;
var f5=false;
var f6=false;
var f7=false;
```

```
var f8=false;
var f9=false;
function Mcheck(){
    if (document.MAIL4.Namae.value=="") {
        window.alert("名前を入力してください");
        return false }
    if (f1==false&&f2==false&&f3==false&&f4==false&&f5==false&&f6
==false&&f7==false&&f8==false&&f9==false) {
        window.alert("チェックボックスを、どれかひとつ以上チェックしてください");
        return false }
     return true;
}
function Frest() { f1=false;f2=false;f3=false;f4=false;f5=false;f6
=false;f7=false;f8=false;f9=false; }
//-->
</script>
 ～中略～
<body>
*フォームの内容をチェックする<p>
<form name="MAIL4" action="mailto:****@*******.ne.jp" method="post
" onSubmit="return Mcheck()">
<b>お名前:</b><input type="text" name="Namae" size=20><br>(必ず入力し
てください)<br><p>
<b>ブラウザは何をお使いですか</b><br>(必ずひとつ以上チェックしてください)<br>
<input type="checkbox" name="NorE1" value="Netscape 7.X" onClick=
"f1=!f1">Netscape 7.X<br>
<input type="checkbox" name="NorE2" value="Netscape 6.X" onClick=
"f2=!f2">Netscape 6.X<br>
<input type="checkbox" name="NorE3" value="Netscape Navigator 4.X"
onClick="f3=!f3">Netscape Navigator 4.X<br>
<input type="checkbox" name="NorE4" value="Firefox" onClick="f4=!f
4">Firefox(Mozilla)<br>
<input type="checkbox" name="NorE5" value="Internet Explorer 6.X"
onClick="f5=!f5">Internet Explorer 6.X<br>
<input type="checkbox" name="NorE6" value="Internet Explorer 5.X"
onClick="f6=!f6">Internet Explorer 5.X<br>
<input type="checkbox" name="NorE7" value="Internet Explorer 4.X"
onClick="f7=!f7">Internet Explorer 4.X<br>
<input type="checkbox" name="NorE8" value="Safari" onClick="f8=!
f8">Safari<br>
<input type="checkbox" name="NorE9" value="other" onClick="f9=!f9"
>その他<p>
<input type="submit" name="BOOK1" value=" メールを送る ">
<input type="reset" value=" リセットする " onClick="Frest()">
</form>
</body>
```

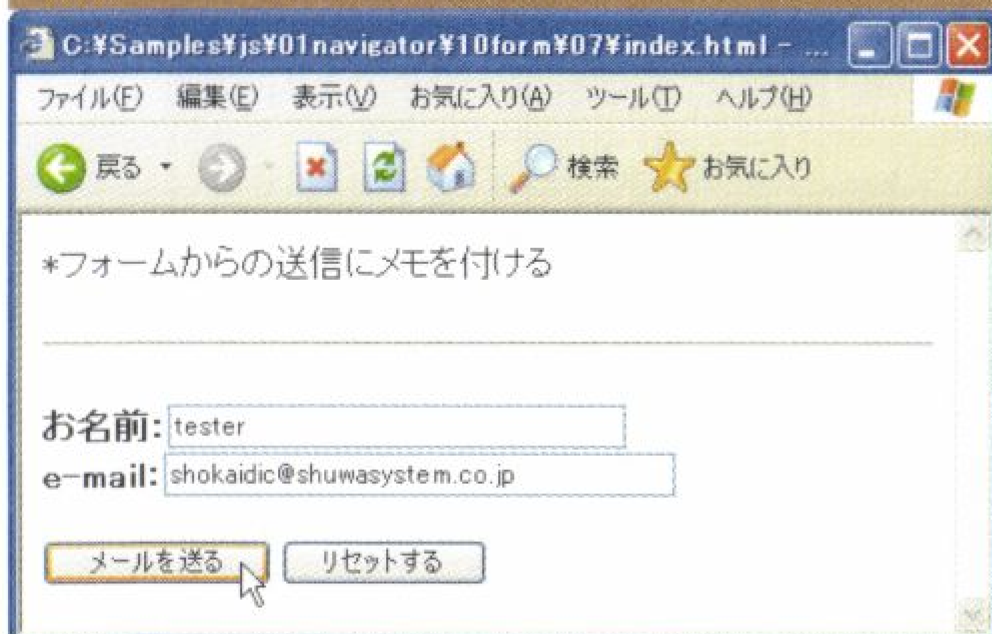
alert()：「windowオブジェクト」の「警告用のダイアログボックスを開く」(P.333)

# フォームからの送信にメモを付ける

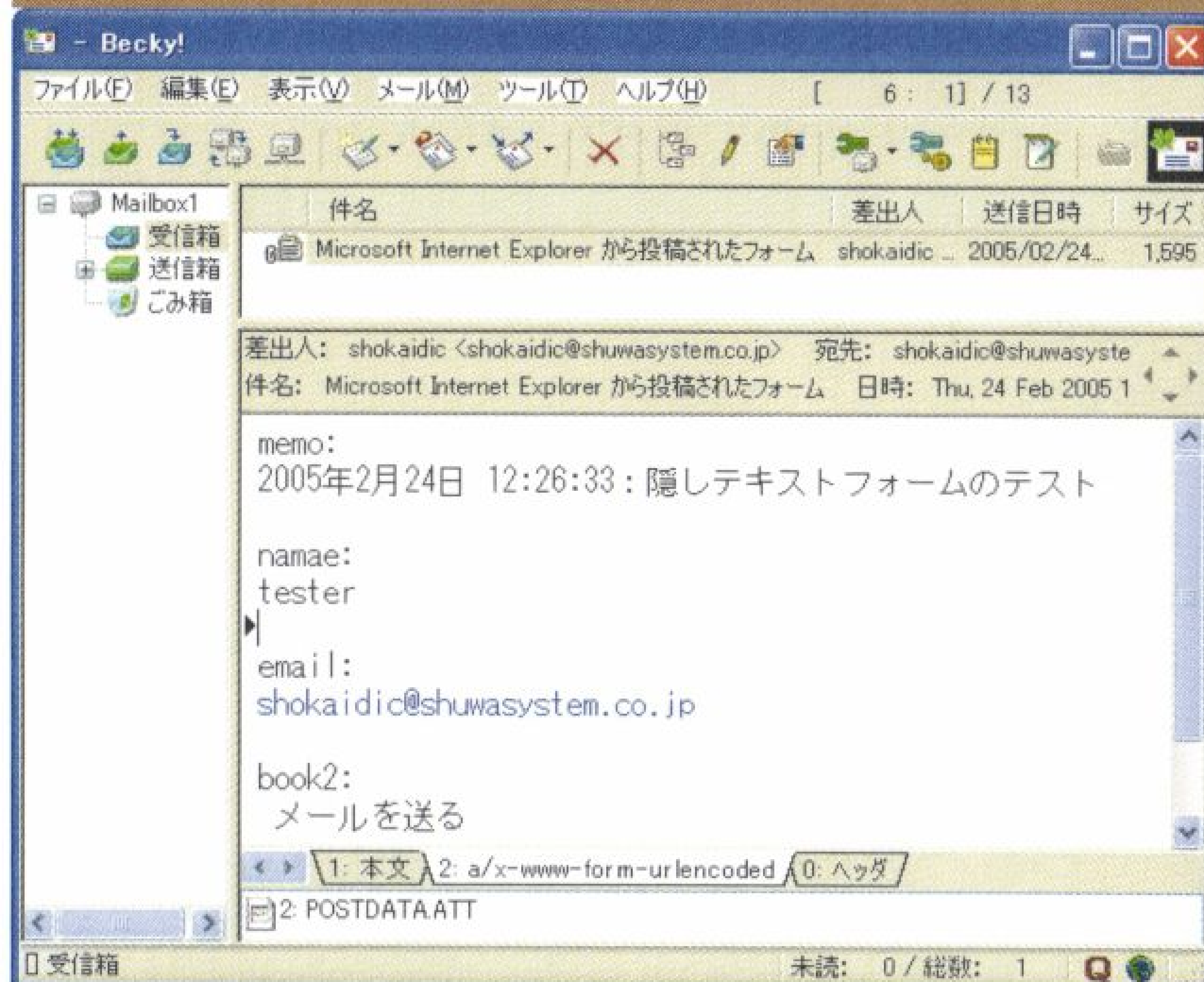
**<input type="hidden"name="*hiddenオブジェクト名*" value="*値*">**
**onSubmit="*スクリプト | 関数*"** [イベントハンドラ]



Before




After




隠しテキストフォームhiddenオブジェクトを使えば、フォームにメモ的な情報を付加することができます。

サンプルでは、[メールを送る]ボタンをクリックした時に、イベントハンドラ「onSubmit」が関数の処理を発生させます。そして、隠しテキストフォームに、「document.title」プロパティで取得したHTMLファイルの「<title></title>」部分とコメント、「now.toLocaleString()」メソッドで取得した日時を付加して送信しています。

また、フォームの内容を「mailto:」で送るようにしているので、サンプルのままでは「mailto:」に対応していないブラウザでフォームの内容を送ることはできませんが、hiddenオブジェクト自体は機能します。

実際に試す場合には、サンプルの「****@******.ne.jp」の部分を、メールを受け取りたいメールアドレスに変更するか、データを受け取るCGIサーバーを設定してください。

## Sample

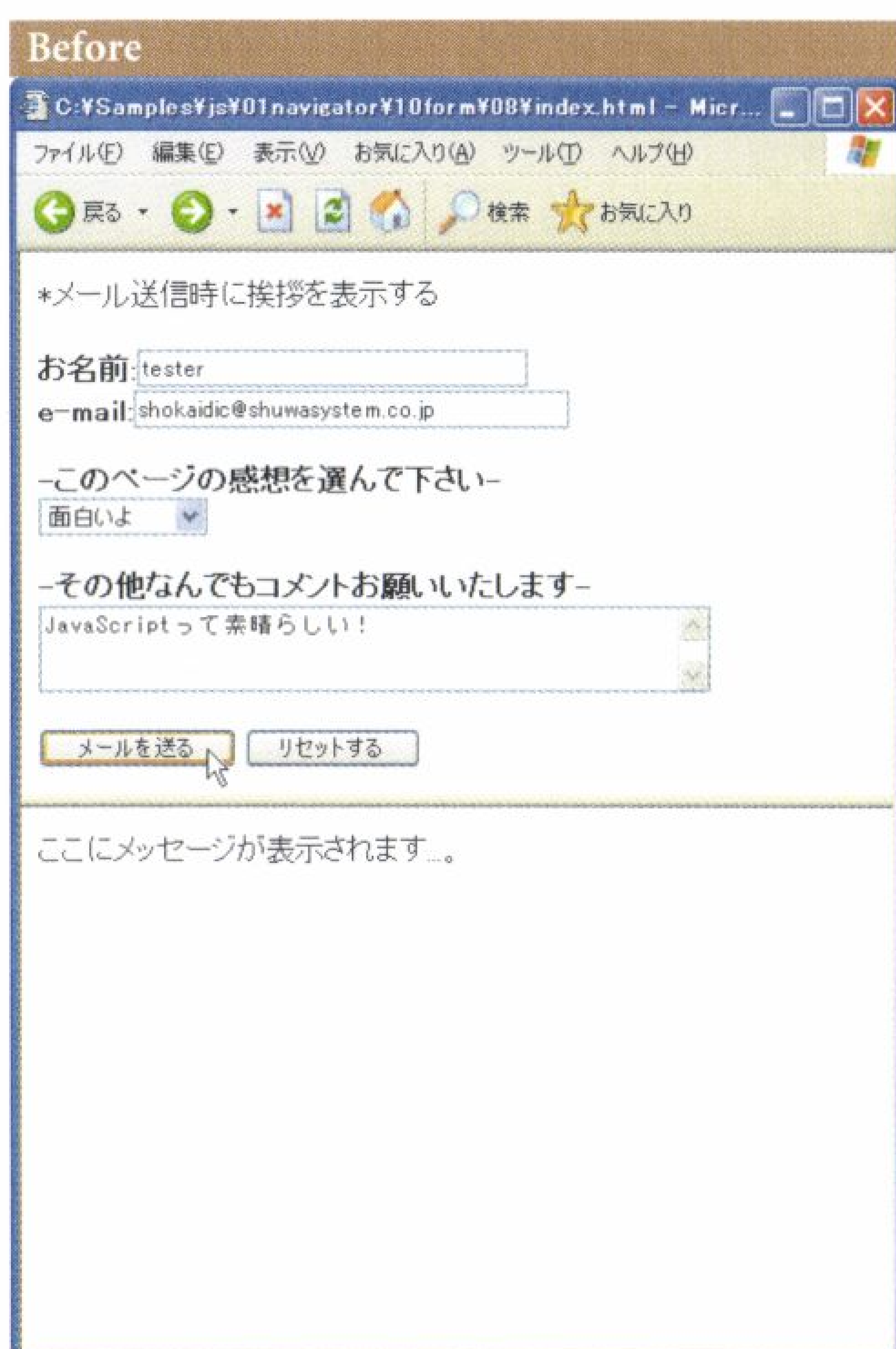
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function Memo(){
    var now = new Date();
    var time = now.toLocaleString();
    var tit = document.title;
    var com = ":隠しテキストフォームのテスト";
    document.MAIL2.memo.value = time + tit + com;
}
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
＊フォームからの送信にメモを付ける
<p>
<hr>
<form name="MAIL2" action="mailto:****@*******.ne.jp" method="post" onsubmit="Memo()">
<input type="hidden" name="memo" value="">
<b>お名前:</b><input type="text" name="namae" size=40><br>
<b>e-mail:</b><input type="text" name="email" size=45><p>
<input type="submit" name= "book2" value=" メールを送る ">
<input type="reset" value=" リセットする ">
</form>
</p>
</body>
</html>
```


<form>:「フォーム」の「フォームを作る」(P.113)
<input type="hidden">:「フォーム」の「表示されないフィールドを作る」(P.123)
mailto:「フォーム」の「フォームの内容がメールで届くようにする」(P.133)
document.title:「documentオブジェクト」の「ドキュメントの情報を取得する」(P.386)
Date():「Dateオブジェクト」の「年・月・日・時・分・秒を表示する」(P.500)
toLocaleString():「Dateオブジェクト」の「国際標準時やローカルタイムを表示する」(P.506)
onSubmit:リファレンス「イベントハンドラ」の「onSubmit」(P.621)

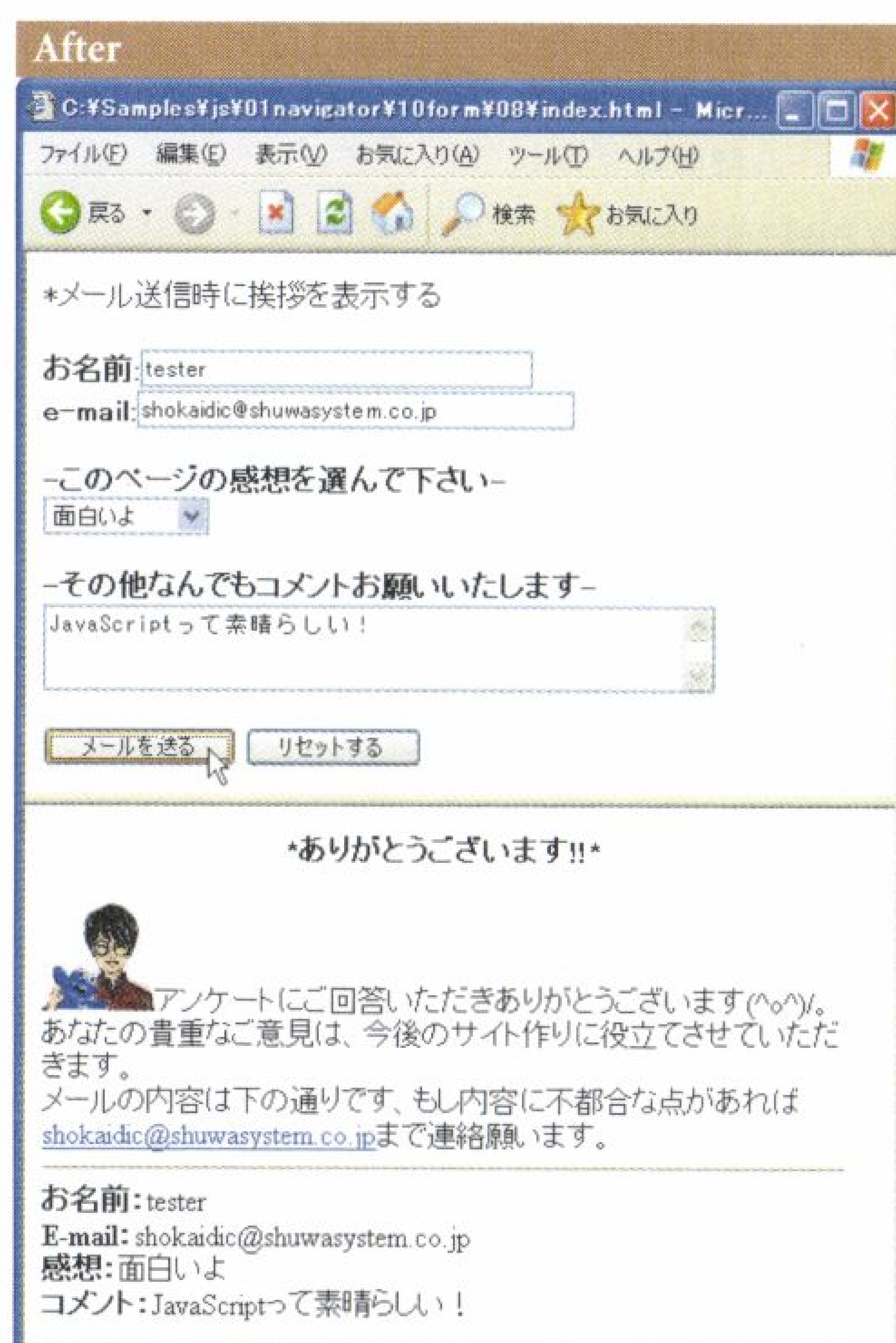
# メール送信時に挨拶を表示する

onSubmit="スクリプト | 関数" [イベントハンドラ]
document.フォームオブジェクト名.オブジェクト名.value
document.フォームオブジェクト名.selectオブジェクト名.options
[document.フォームオブジェクト名.selectオブジェクト名.selected
Index].value

Before




After




サンプルでは、[メールを送る]ボタンが押されると、メールを送る動作と同時に、フレームにお礼とメールの内容を表示するようにしています。
CGIなどでは、送信されたデータをサーバーが正常に受け取った後で内容確認の文章を表示します。しかしJavaScriptの場合は、イベントハンドラ「onSubmit」が[メールを送る]ボタンが押された時にJavaScriptの処理を実行させるので、送信が正常に行われたかどうかの判断はできないので、注意してください。
フォーム内の各値の書き出しは、textオブジェクトは「document.フォームオブジェクト名.オブジェクト名text.value」、optionオブジェクトは「document.フォームオブジェクト名.selectオブジェクト名.options[document.フォームオブジェクト名.selectオブジェクト名.selectedIndex].value」としています。各オブジェクトの階層関係には、十分に注意してください。
実際に試す場合には、この他にも「h2.html」を用意し、サンプルの「****@******.ne.jp」の部分を、メールを受け取りたいメールアドレスに変更するか、データを受け取るCGIサーバーを設定してください。

## Sample

**【フレームウィンドウ】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" >
<html>
<head>
<title></title>
</head>
<frameset rows="*,300">
    <frame src="f1.html" name="f1">
    <frame src="f2.html" name="f2">
</frameset>
<noframes>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</noframes>
</html>
```

**【f1.html】**

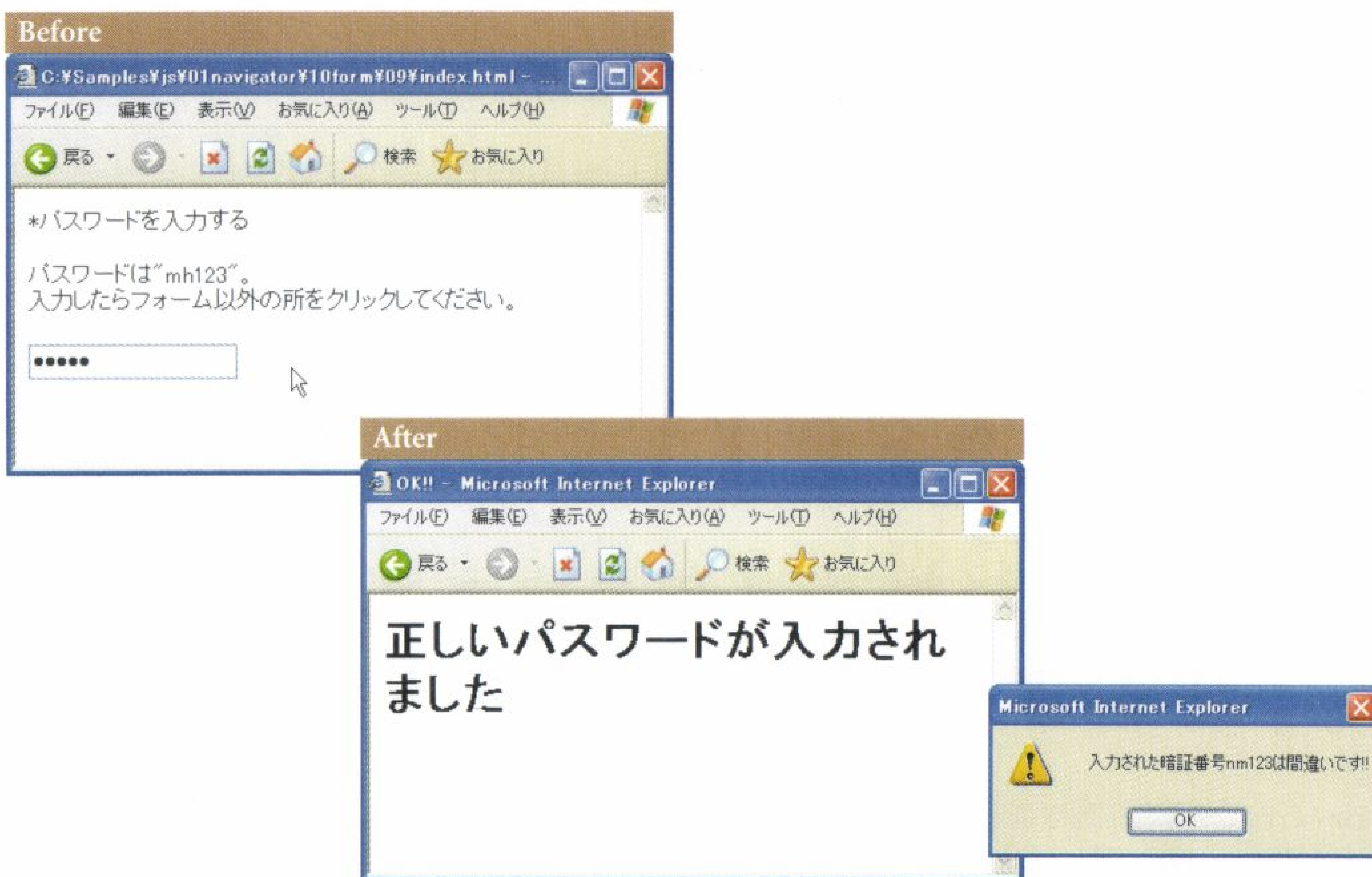
```
<script type="text/javascript">
<!--
function Mopen(){
    parent.f2.document.open();
    parent.f2.document.write("<html>");
    parent.f2.document.write("<head><title>ArigatouGozaimas</title
></head>");
    parent.f2.document.write("<body>");
    parent.f2.document.write("<center><b>*ありがとうございます!!*</b></
center>");
    parent.f2.document.write("<br>");
    parent.f2.document.write("<img src='image.jpg' align='image.jp
g'>");
    parent.f2.document.write("アンケートにご回答いただきありがとうございます(
^o^)/。");
    parent.f2.document.write("<br>");
    parent.f2.document.write("あなたの貴重なご意見は、今後のサイト作りに役立て
させていただきます。");
    parent.f2.document.write("<br>");
    parent.f2.document.write("メールの内容は下の通りです、もし内容に不都合な点
があれば");
    parent.f2.document.write("<br>");
    parent.f2.document.write("<a href='mailto:****@*******.ne.jp'>
****@*******.ne.jp</a>");
    parent.f2.document.write("まで連絡願います。");
    parent.f2.document.write("<hr>");
    parent.f2.document.write("お名前:".bold());
    parent.f2.document.write(document.MAIL1.Namae.value);
    parent.f2.document.write("<br>");
    parent.f2.document.write("E-mail:".bold());
    parent.f2.document.write(document.MAIL1.EMail.value);
```

```
    parent.f2.document.write("<br>");
    parent.f2.document.write("感想:".bold());
    parent.f2.document.write(document.MAIL1.Kansou.options[document.MAIL1.Kansou.selectedIndex].value);
    parent.f2.document.write("<br>");
    parent.f2.document.write("コメント:".bold());
    parent.f2.document.write(document.MAIL1.Comment.value);
    parent.f2.document.write("</body>");
    parent.f2.document.write("</html>");
    parent.f2.document.close();
     }
//-->
</script>
 ～中略～
</head>
<body>
*メール送信時に挨拶を表示する
<form name="MAIL1" action="mailto:****@*******.ne.jp" method="post" onSubmit="Mopen()">
<p>
<b>お名前:</b><input type="text" name="Namae" size=40><br>
<b>e-mail:</b><input type="text" name="EMail" size=45>
</p>
<p>
<b>-このページの感想を選んで下さい-</b><br>
<select name="Kansou" >
    <option value="大変面白い">大変面白い
    <option value="面白いよ">面白いよ
    <option value="ふつうだな">ふつうだな
    <option value="つまんないよ">つまんないよ
    <option value="よくわからん!!">よくわからん!!
</select>
</p>
<p>
<b>-その他なんでもコメントお願いいたします-</b><br>
<textarea name="Comment" rows=3 cols=50>
</textarea>
</p>
<p>
<input type="submit" name="BOOK1" value=" メールを送る ">
<input type="reset" value=" リセットする ">
</p>
</form>
</body>
```


parent:「frameオブジェクト」の「入力されたURLを別フレームに表示する」(P.374)
document.open()・document.close():「documentオブジェクト」の「開いたウィンドウに文字を記述する」(P.388)
bold():「stringオブジェクト」の「太字(ボールド)にする」(P.547)

# パスワードを入力する

**<input type="password" name="*passwordオブジェクト名*" value="*値*" *イベントハンドラ*>**




passwordオブジェクトは、一見すると普通のテキストの入力欄と同じですが、このフォーム内に入力された値は、黒丸などの他からはわからないものに置き換えられて表示されます。
サンプルでは、パスワードを入力してもらい、それが正しければ別のページをロードし、間違っていれば警告を出します。JavaScriptは基本的にソースコードを隠すようにはできていませんので、パスワードを完全に隠匿することは難しく、passwordオブジェクトを実際に使う場面としては、CGIの入力用インターフェイスとして使用することを前提としていると思われます。
なお、passwordオブジェクトはJavaScript1.0からのJavaScriptですが、Netscape Navigator 2.0では値を正確に取得できない場合があります。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function  GetP(s) {
```

```
    if (s=="mh123")  { location.href = "OK.html" }
     else { alert("入力された暗証番号"+s+"は間違いです!!") }
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*パスワードを入力する
<p>
パスワードは"mh123"。<br>
入力したらフォームの以外の所をクリックしてください。<br>
<form name="ANSHYO">
<input type="password" name="anshyo" onBlur=" GetP(this.value)"
value="">
</form>
</p>
</body></html>
```


<input type="password">:「フォーム」の「パスワードの入力フィールドを作る」(P.122)
window.alert():「windowオブジェクト」の「警告用のダイアログボックスを開く」(P.333)
onBlur:リファレンス「イベントハンドラ」の「onBlur」(P.619)

**TIPS**

## JavaScriptで「"」を書き出す方法

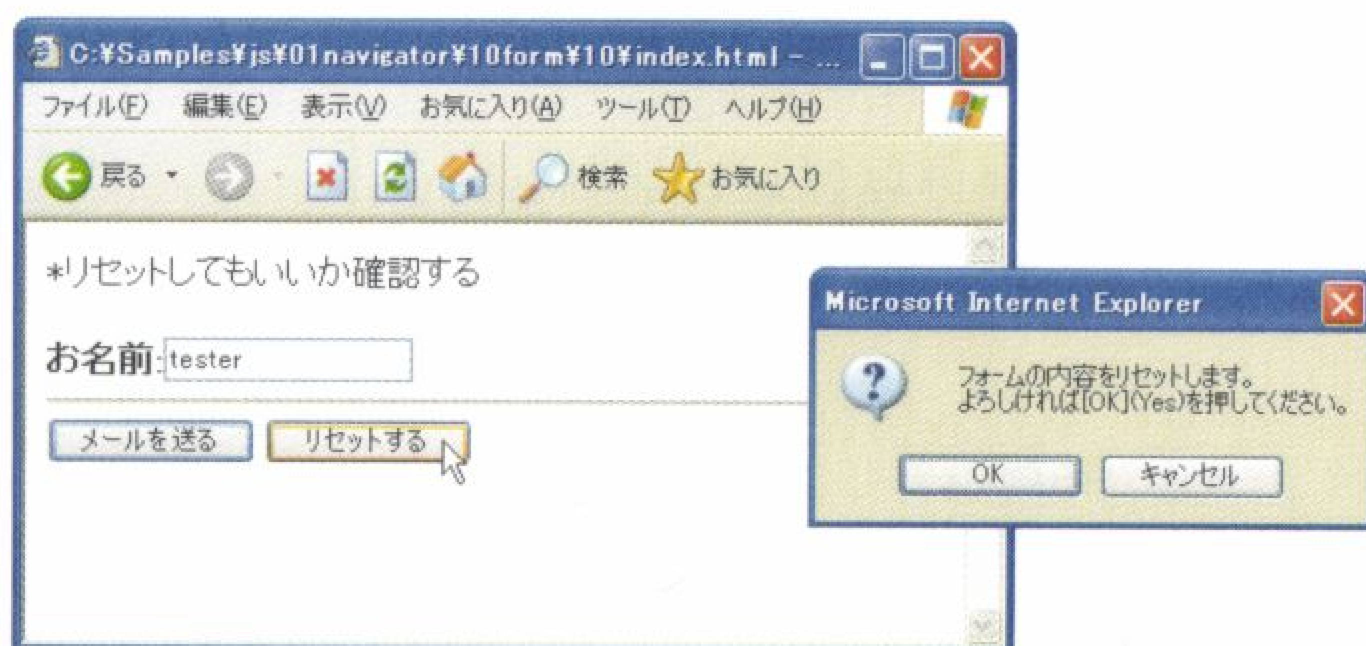
JavaScriptでは、文字列をダブルクォーテーションマーク「"」で囲んで指定する必要があり、その中では「"」を使うことはできません。

では、JavaScriptでダブルクォーテーションマーク「"」を書き出したい時には、どうすればよいのでしょうか?

まずひとつの方法として、特殊キャラクター文字の「¥"」を使う方法があります。そして、もうひとつは、サンプルのように「"」と「'」のネスト(入れ子)の関係を逆にして、文字列をシングルクォーテーションマーク「'」で囲み、その中で「"」を使うという方法があります。

# リセットしてもいいか確認する

**onReset="スクリプト | 関数"** [イベントハンドラ]




イベントハンドラ「onReset」は、リセットボタンが押された時のイベントを取得します。サンプルでは、[リセットする]のボタンが押された時に確認のダイアログボックスを開き、[OK(Yes)]が押された時はフォームをリセットし、そうでない時はリセットの処理を中止しています。
JavaScript1.1で追加されたイベントハンドラです。
実際に試す場合には、サンプルの「****@******.ne.jp」の部分を、メールを受け取りたいメールアドレスに変更するか、データを受け取るCGIサーバーを設定してください。

## Sample

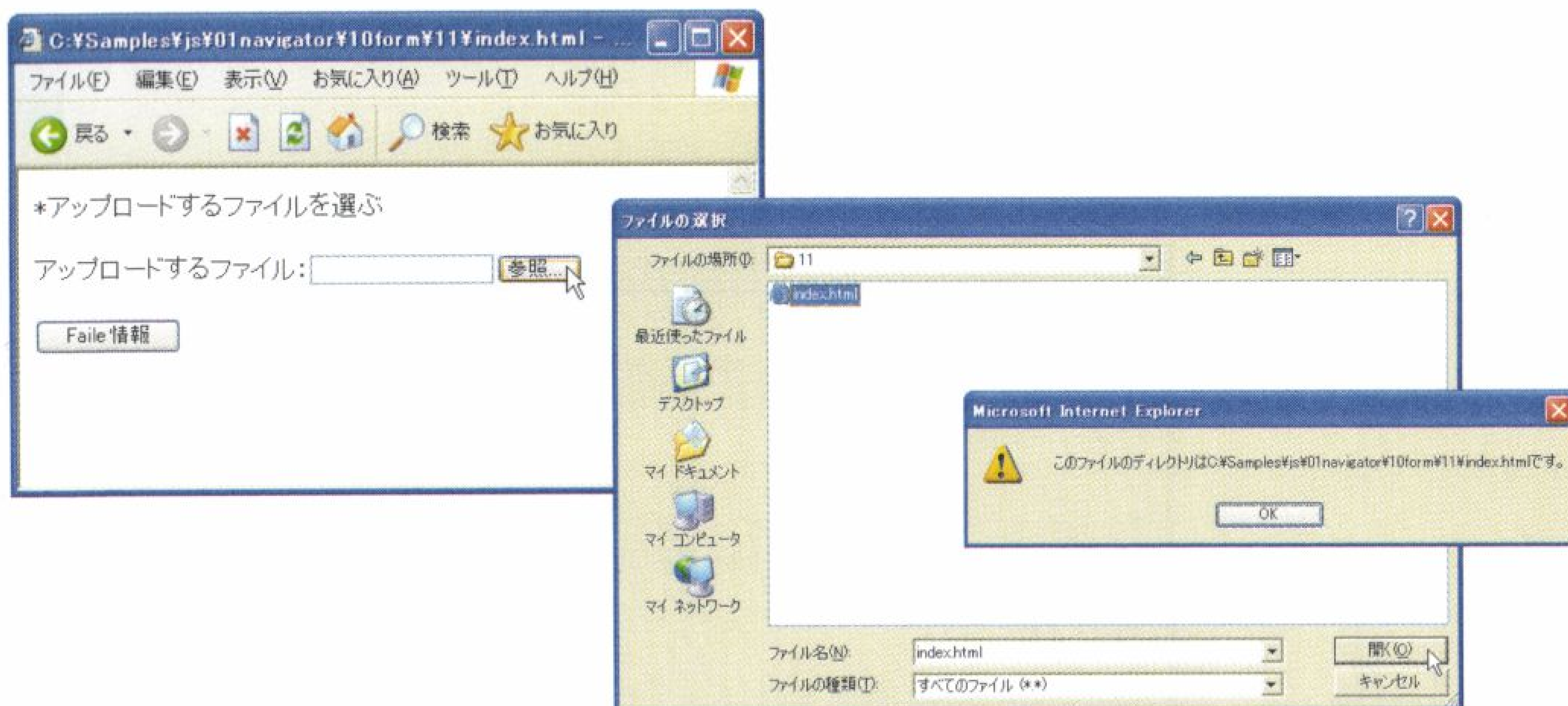
```
<script type="text/javascript">
<!--
function Mcheck() {
    if ( confirm ('フォームの内容をリセットします。¥nよろしければ[OK](Yes)を押してください。')) { return true }
return false }
//-->
</script>
  ～中略～
<form name="MAIL4" action="mailto:****@******.ne.jp" method="post" onReset="return Mcheck()">
<b>お名前:</b><input name="Namae" size=20><br>
<hr>
<input type="submit" name="BOOK1" value=" メールを送る ">
<input type="reset" value=" リセットする ">
</form>
</p>
</body>
```


confirm():「windowオブジェクト」の「確認ボタン付きのダイアログボックスを開く」(P.334)

# アップロードするファイルを選ぶ

**<input type="file" name="fileオブジェクト名">**
**document.フォーム名.fileオブジェクト名.value** [プロパティ]






FileUploadオブジェクトは、アップロードするファイルを選択するフォームです。[参照...]ボタンをクリックするとローカルのディレクトリが参照でき、選択するとテキスト欄にファイル名が表示されます。その時に、「value」プロパティにはディレクトリの情報が納められ、「document.フォーム名.fileオブジェクト名.value」で参照することができます。
FileUploadオブジェクトは、あくまでもアップロードするファイルを選択することしかできず、実際にアップロードする作業は、HTMLやCGIの力を借りることになります。これは、JavaScriptがセキュリティ確保のため、ローカルのリソースにアクセスすることを厳しく禁止しているためです。
JavaScript1.1で追加されたオブジェクトです。

## Sample

```
<form name="form1">
<p>
アップロードするファイル: <input type="file" name="UploadFile">
</p>
<input type="button" value=" Faile情報 "
    onClick="alert('このファイルのディレクトリは' + document.form1.Upload
File.value+'です。')">
</form>
```

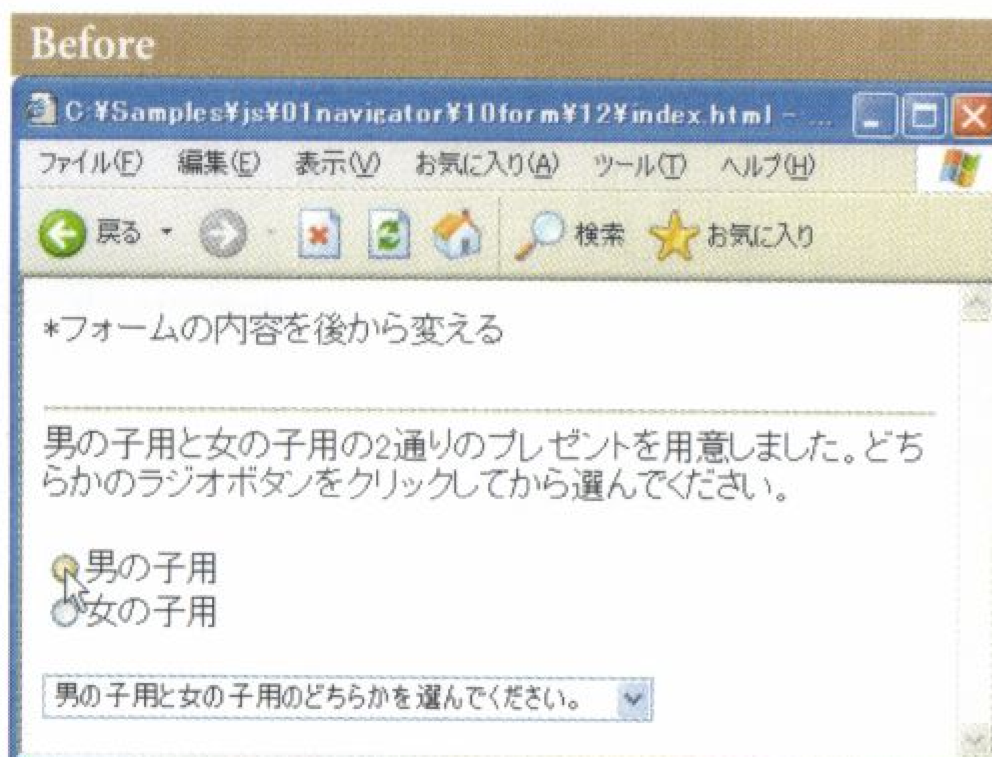

<input type="file">:「フォーム」の「ファイル選択の機能を付ける」(P.130)
alert():「windowオブジェクト」の「警告用のダイアログボックスを開く」(P.333)

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

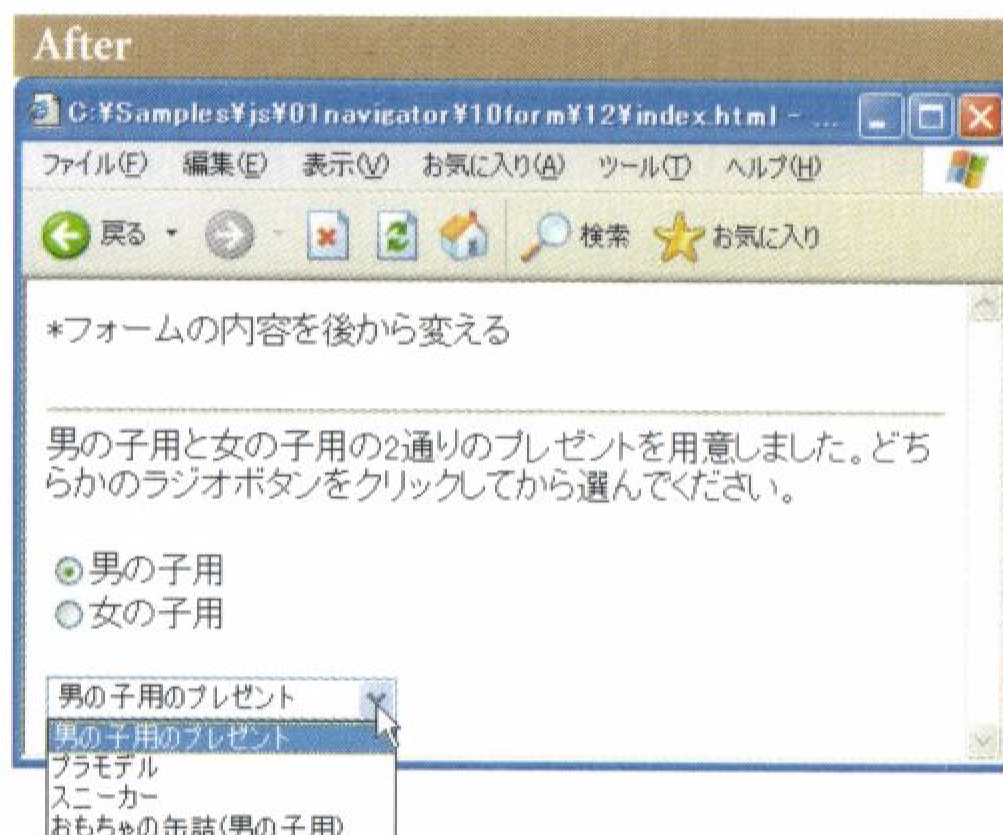
# フォームの内容を後から変える

**<input type="radio" name="*radioオブジェクト名*" value="*値*" *イベントハンドラ*>**
**<select name="*selectオブジェクト*">**
**<option>**
**options[*インデックス*]**

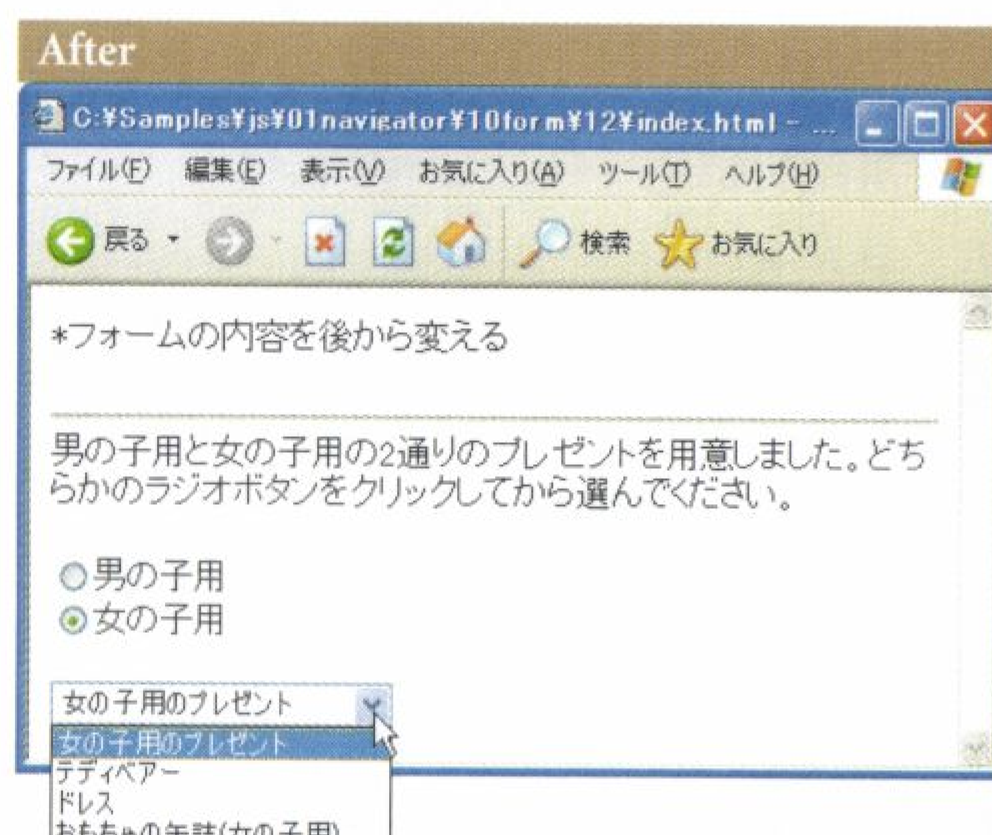
Before




After




After




JavaScript1.1からは、フォームの内容を後から変更できるようになりました。サンプルでは、「options[インデックス]」で[男の子用]と[女の子用]の2種類のoptionオブジェクトの配列を作り、それをラジオボタンのクリックで切り替えています。JavaScript1.1で追加されたプロパティです。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
```

```
<script type="text/javascript">
<!--
function BY(PR) {
    PR.pr.options[0].text = "男の子用のプレゼント";
    PR.pr.options[1].text = "プラモデル";
    PR.pr.options[2].text = "スニーカー";
    PR.pr.options[3].text = "おもちゃの缶詰(男の子用)";
}
function GR(PR) {
    PR.pr.options[0].text = "女の子用のプレゼント";
    PR.pr.options[1].text = "テディベアー";
    PR.pr.options[2].text = "ドレス";
    PR.pr.options[3].text = "おもちゃの缶詰(女の子用)";
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

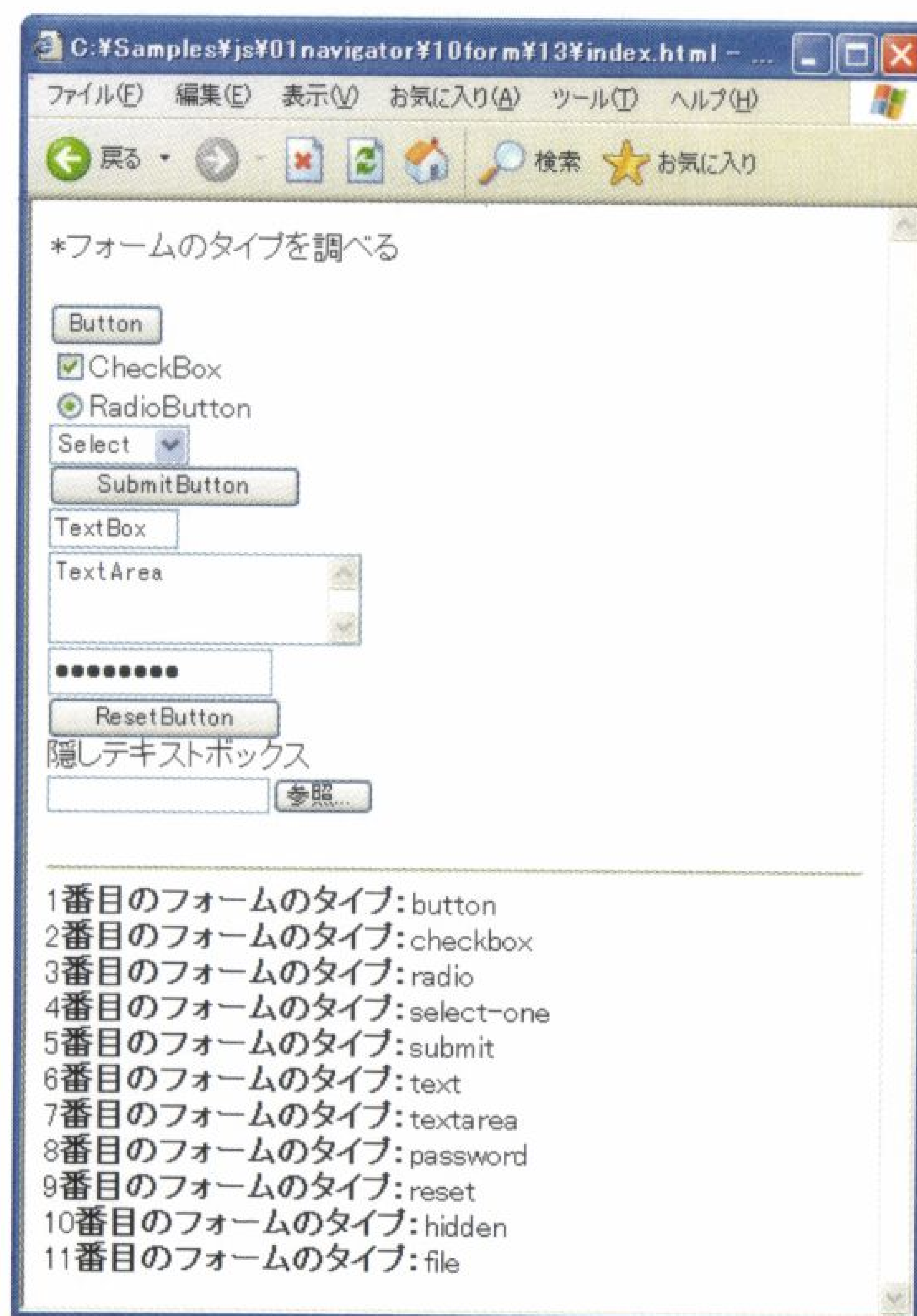
</head>
<body>
*フォームの内容を後から変える
<p>
<hr>
男の子用と女の子用の2通りのプレゼントを用意しました。どちらかのラジオボタンをクリックして
から選んでください。
</p>
<form name="BORG">
    <input type="radio" name="borg" value="BOY" onClick="BY(this.
form)">男の子用
<br>
    <input type="radio" name="borg" value="GIR" onClick="GR(this.
form)">女の子用
<p>
<select name="pr">
<option>男の子用と女の子用のどちらかを選んでください。
<option>----------------------------------------
<option>----------------------------------------
<option>----------------------------------------
</select>
</form>
</p>
</body></html>
```


<input type="radio">：「フォーム」の「ラジオボタンを作る」(P.124)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# フォームのタイプを調べる

**document.フォームオブジェクト名.elements[*i*].type** ［プロパティ］




「type」プロパティは、「button」や「checkbox」などのフォームのタイプを返すプロパティです。
サンプルでは、フォームオブジェクト「FTYPE」内の各フォーム関連のオブジェクトを配列として捉え、上から順番にタイプを書き出しています。
JavaScript1.1で追加されたプロパティです。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
```

```
</head>
<body>
*フォームのタイプを調べる
<p>
<form name="FTYPE">
    <input type="button" name="BUTTON" value="Button"><br>
    <input type="checkbox" name="CHECBOX" value="CheckBox" checked
>CheckBox<br>
    <input type="radio" name="RADIO" value="RadioButton" checked>R
adioButton<br>
    <select name="SELECT">
        <option>Select
        <option>Select1
    </select><br>
    <input type="submit" name="SUBMIT" value=" SubmitButton "><br>
    <input type="text" name="TEXT" value="TextBox" size=10 ><br>
    <textarea name="TEXTAREA" rows=3 cols=20 >TextArea</textarea><
br>
    <input type="password" name="PASSWORD" value="Password" size=1
5><br>
    <input type="reset" name="RESET" value=" ResetButton "><br>
    <input type="hidden" name="HIDDEN" value="Hidden">隠しテキストボッ
クス<br>
    <input type="file" name="UploadFile">
</form>
</p>
<hr>

<script type="text/javascript">
<!--
var L = document.FTYPE.elements.length;
for(i=0; i<L; i++){
document.write( i+1 + "番目のフォームのタイプ:".bold());
document.write(document.FTYPE.elements[i].type);
document.write("<br>");
}
//-->
</script>

</body>
</html>
```

bold():「stringオブジェクト」の「太字(ボールド)にする」(P.547)

# マップエリア外をクリックしたら警告用のウィンドウを開く

**href="javascript:関数"**




サンプルでは、イメージマップのリンクを指定した範囲内がクリックされた時は「JavaScript:」で関数を発生させページを読み込み、範囲外をクリックされた時は警告用のダイアログボックスを開くようにしています。
このサンプルスクリプトは、Netscape Navigator 2.X でも使用できます。
実際に試す場合には、この他にも「MAP1.html」~「MAP3.html」の3つのファイルを用意してください。

## Sample

**【メインウィンドウ】**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function Go(url){ location.href=url }
function Miss() { alert("イメージマップのエリア外です!!") }
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
```

```
</head>
<body>
＊マップエリア外をクリックしたら警告用のウィンドウを開く
<p>
<map name="ARIA1">
    <area name="aria1" shape=rect coords=13,9,73,69 href="JavaScri
pt:Go('MAP1.html')">
    <area name="aria2" shape=circle coords=119,38,29 href="JavaScr
ipt:Go('MAP2.html')">
    <area name="aria3" shape=poly coords=160,69,188,7,222,69 href=
"JavaScript:Go('MAP3.html')">
    <area name="aria4" shape=rect coords=0,0,234,81 href="JavaScri
pt:Miss()">
</map>
<img src="MAP.jpg" Usemap="#ARIA1" border=0 width="234" height="81"
alt="ARIATEST">
</p>
</body>
</html>
```

<map><area>：「画像とマルチメディア」の「イメージマップを作成する」(P.103)
alert()：「windowオブジェクト」の「警告用のダイアログボックスを開く」(P.333)
focus()：「windowオブジェクト」の「ウィンドウを前に出す」(P.345)
location.href：「locationオブジェクト」の「自ページのURLを取得する」(P.408)

**TIPS**

## エリアマップ内でNetscape Navigator 2.0でも使えるJavaScriptは?

Areaオブジェクトは、JavaScript1.1から追加されたオブジェクトですが、<area>内で「href="JavaScript:関数"」を使えば、JavaScript1.1に対応していないNetscape Navigator 2.0でも機能するスクリプトを作ることができます。これは、Areaオブジェクトがlinkの配列内にあるためだと思われます。

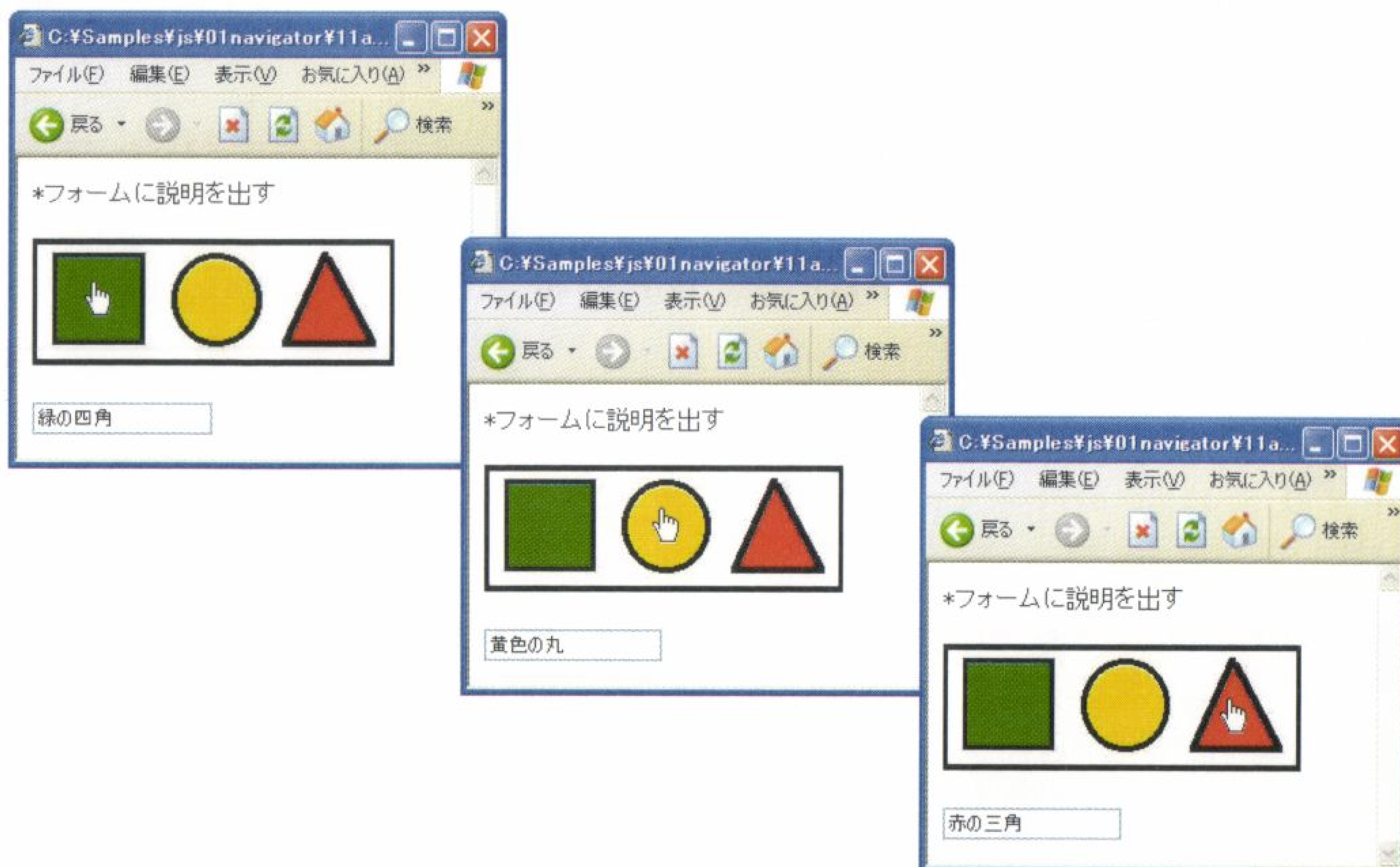
つまり、Netscape Navigator 2.0では、<area>内の「href="JavaScript:関数"」は、リンクオブジェクトとして捉えられているのでしょう。

しかし、<area>に記された「onMouseOver」や「onMouseOut」などのイベントハンドラは、Netscape Navigator 2.0からサポートされているJavaScript1.0に関しても、評価はされません。

したがって、たとえば「フォームに説明を出す」(次ページ)や「イメージマップをリンク以外の機能で使う」(P.452)のサンプルは、Netscape Navigator 2.0で実行すると、フォームに文字を書き出すスクリプトの部分は動きません。ただし、新しいウィンドウを開いたり、背景色を変えたりする部分のスクリプトは、正常に動きます。

# フォームに説明を出す

**href="javascript:*関数*"**
**onMouseOver="*スクリプト | 関数*"** [イベントハンドラ]
**onMouseOut="*スクリプト | 関数*"** [イベントハンドラ]




サンプルでは、指定されたエリア内にマウスカーソルが乗った時に、イベントハンドラ「onMouseOver」がフォームに文字を書き出します。エリア内からマウスカーソルが離れた時は、イベントハンドラ「onMouseOut」が関数「MessCr()」を発生させてフォームに「" "」の値を引き渡し、それによってフォーム内の文字を消去しています。
イベントハンドラ「onMouseOver」は、Netscape Navigator 2.Xから使用可能なJava Script1.0に対応しているイベントハンドラですが、<area>内では評価されません。
なお、リンクをクリックすると新しいページが開きます。
「onMouseOut」は、JavaScript1.1で追加されたイベントハンドラです。
実際に試す場合には、この他にも「MAP2.html」と「MAP3.html」を用意してください。

## Sample

**【メインウィンドウ】**

```
<script type="text/javascript">
<!--
function Go(url) { window.open( url ,"IWindow" ) }
function Mis() { alert("イメージマップのエリア外です!!") }
function MessCr() { document.Fmess.fmess.value = "" }
//-->
</script>
  ～中略～
```

```
</head><body>
*フォームに説明を出す
<p>
<map name="ARIA1">
    <area shape=rect coords=13,9,73,69
href="JavaScript:Go('MAP1.html')"
     onMouseOver="document.Fmess.fmess.value = '緑の四角'"
     onMouseOut="MessCr()">
    <area shape=circle coords=119,38,29 href="JavaScript:Go('MAP2
.html')"
     onMouseOver="document.Fmess.fmess.value = '黄色の丸'"
     onMouseOut="MessCr()">
    <area shape=poly coords=160,69,188,7,222,69 href="JavaScript:
Go('MAP3.html')"
     onMouseOver="document.Fmess.fmess.value = '赤の三角'"
     onMouseOut="MessCr()">
    <area shape=rect coords=0,0,234,81 href="JavaScript:Mis()"
     onMouseOver="MessCr()">
</map>
<img src="MAP.jpg" usemap="#ARIA1" border=0 width="234" height="81
" alt="ARIATEST">
</p>
<p>
<form name="Fmess">
<input type="text" name="fmess" size=20>
</form>
</p>
</body></html>
```

**【MAP1.html】**

```
<script type="text/javascript">
<!--
focus()
//-->
</script>
 ～中略～
<body>
<img src="MAP1.jpg" width="60" height="59" alt="MidoriSikaku">
<hr>
<b>*緑の四角*</b>
</body>
```


<map>：「画像とマルチメディア」の「イメージマップを作成する」(P.103)
alert()：「windowオブジェクト」の「警告用のダイアログボックスを開く」(P.333)
window.open()：「windowオブジェクト」の「新しいウィンドウを開く」(P.338)
focus()：「windowオブジェクト」の「ウィンドウを前に出す」(P.345)
location.href：「locationオブジェクト」の「自ページのURLを取得する」(P.408)
<form>：「formオブジェクト」の「フォームに文字を流す」(P.430)

# イメージマップをリンク以外の機能で使う

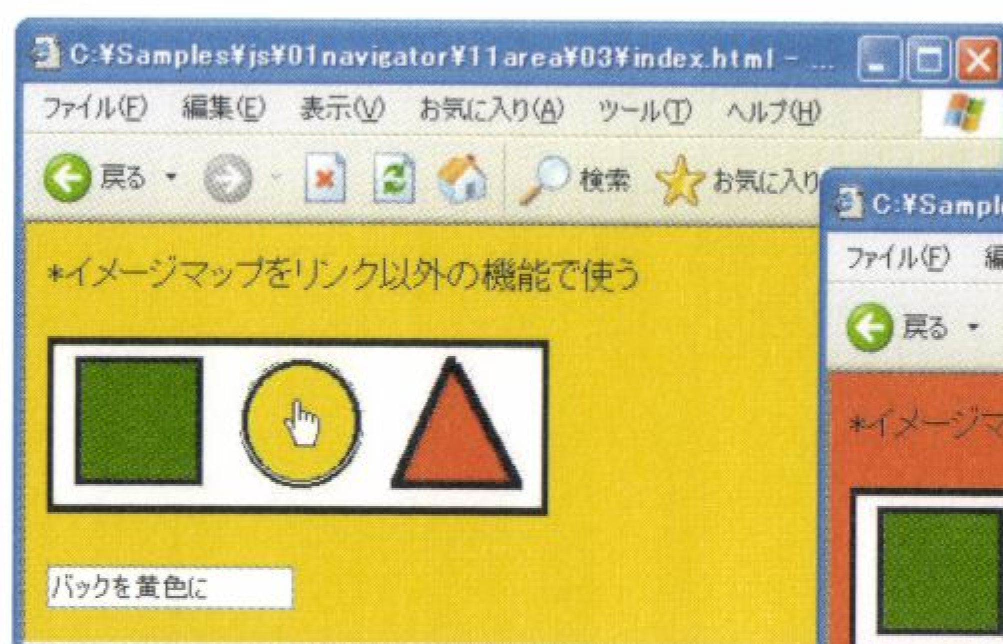
**href="javascript:*関数*"**
**onMouseOver="*スクリプト | 関数*"** [イベントハンドラ]
**onMouseOut="*スクリプト | 関数*"** [イベントハンドラ]

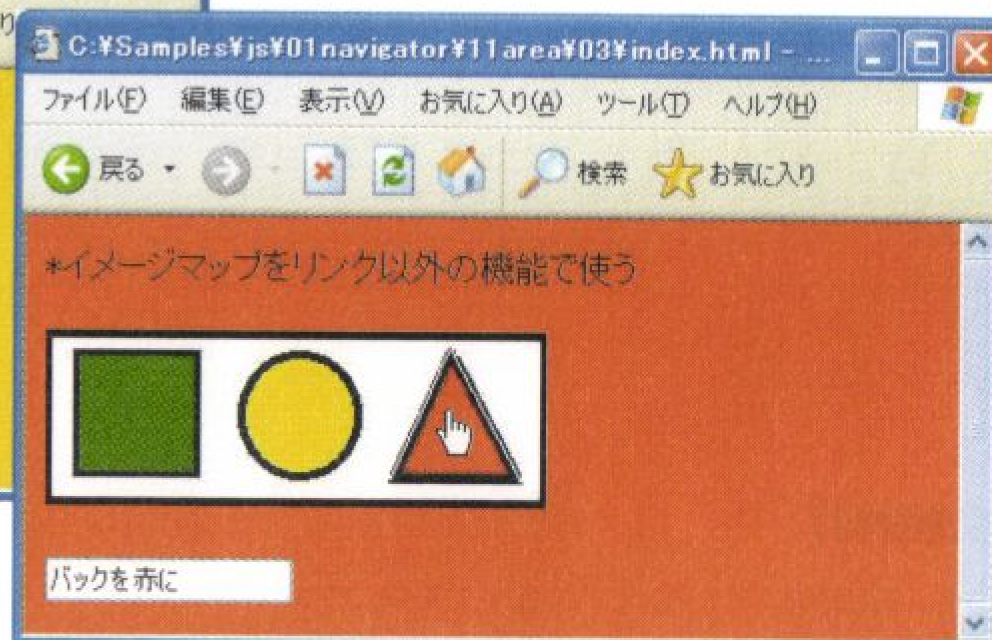
Before




After




サンプルでは、指定されたエリアをクリックすると、「JavaScript:」が関数「BdColor()」を発生させて、中で指定している色の値を「document.bgColor」に引き渡し、背景色を変更しています。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<title></title>

<script type="text/javascript">
<!--
function BdColor(BC) { document.bgColor=BC }
function Mis() { alert("イメージマップのエリア外です!!") }
```

```
function MessCr() { document.Fmess.fmess.value = "" }
//-->
</script>

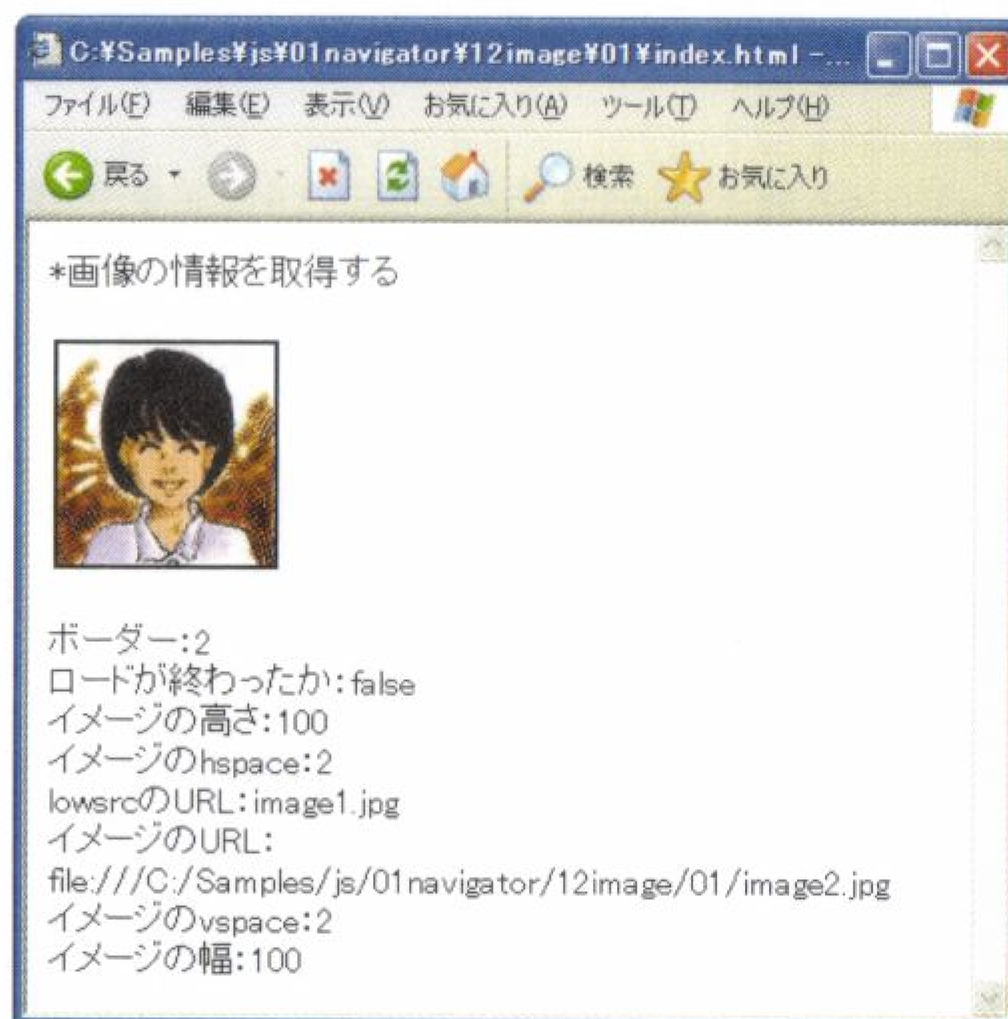
</head>
<body>
*イメージマップをリンク以外の機能で使う
<p>
<map name="ARIA1">
    <area name="aria1" shape=rect coords=13,9,73,69 href="JavaScr
ipt:BdColor('green')"
     onMouseOver="document.Fmess.fmess.value = 'バックを緑に'"
     onMouseOut="MessCr()">
    <area name="aria2" shape=circle coords=119,38,29 href="JavaScr
ipt:BdColor('yellow')"
     onMouseOver="document.Fmess.fmess.value = 'バックを黄色に'"
     onMouseOut="MessCr()" >
    <area name="aria3" shape=poly coords=160,69,188,7,222,69 href=
"JavaScript:BdColor('red')"
     onMouseOver="document.Fmess.fmess.value = 'バックを赤に'"
     onMouseOut="MessCr()">
    <area name="aria4" shape=rect coords=0,0,234,81 href="JavaScri
pt:Mis()"
     onMouseOver="MessCr()">
</map>
<img src="MAP.jpg" usemap="#ARIA1" border=0 width="234" height="81"
alt="ARIATEST">
</p>
<p>
<form name="Fmess">
<input type="text" name="fmess" size=20>
</form>
</p>
</body>
</html>
```


<map>：「画像とマルチメディア」の「イメージマップを作成する」(P.103)
alert()：「windowオブジェクト」の「警告用のダイアログボックスを開く」(P.333)
document.bgColor：「documentオブジェクト」の「背景色を変えるボタンを作る」(P.395)
<input type="text">：「formオブジェクト」の「フォームに文字を流す」(P.430)

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

# 画像の情報を取得する

| | |
|---|---|
| document.オブジェクト名.border | [プロパティ] |
| document.オブジェクト名.complete | [プロパティ] |
| document.オブジェクト名.height | [プロパティ] |
| document.オブジェクト名.hspace | [プロパティ] |
| document.オブジェクト名.lowsrc | [プロパティ] |
| document.images[インデックス].src | [プロパティ] |
| document.images[インデックス].vspace | [プロパティ] |
| document.images[インデックス].width | [プロパティ] |




Imageオブジェクトは、ページ上の画像の0から始まる配列を作成します。Imageオブジェクトの情報は、<img>内で設定した「name」で参照する以外に、配列でも参照できます。

「border」プロパティはボーダーの値を、「complete」プロパティは画像のロードが終っていれば「true」の値を、終っていなければ「false」の値を、「height」プロパティは画像の高さの値を、「hspace」プロパティはドキュメントとの横方向の間隔の値を、「lowsrc」プロパティは正式な画像を表示する前に代わりに表示する低解像度の画像のURLを、「src」プロパティは画像ファイルのURLを、「vspace」プロパティはドキュメントとの縦方向の間隔の値を、「width」プロパティは画像の幅の値を、それぞれ持っています。

「src」プロパティ以外は読み込み専用で、後から変更することはできません。しかし「src」プロパティの値は変更可能なので、それを変えることによってイメージを置き換えることができます。

JavaScript1.1で追加されたオブジェクトです。

なお、このスクリプトでは、Netscape 6.Xで「border」「hspace」「vspace」の値を取得することができません。

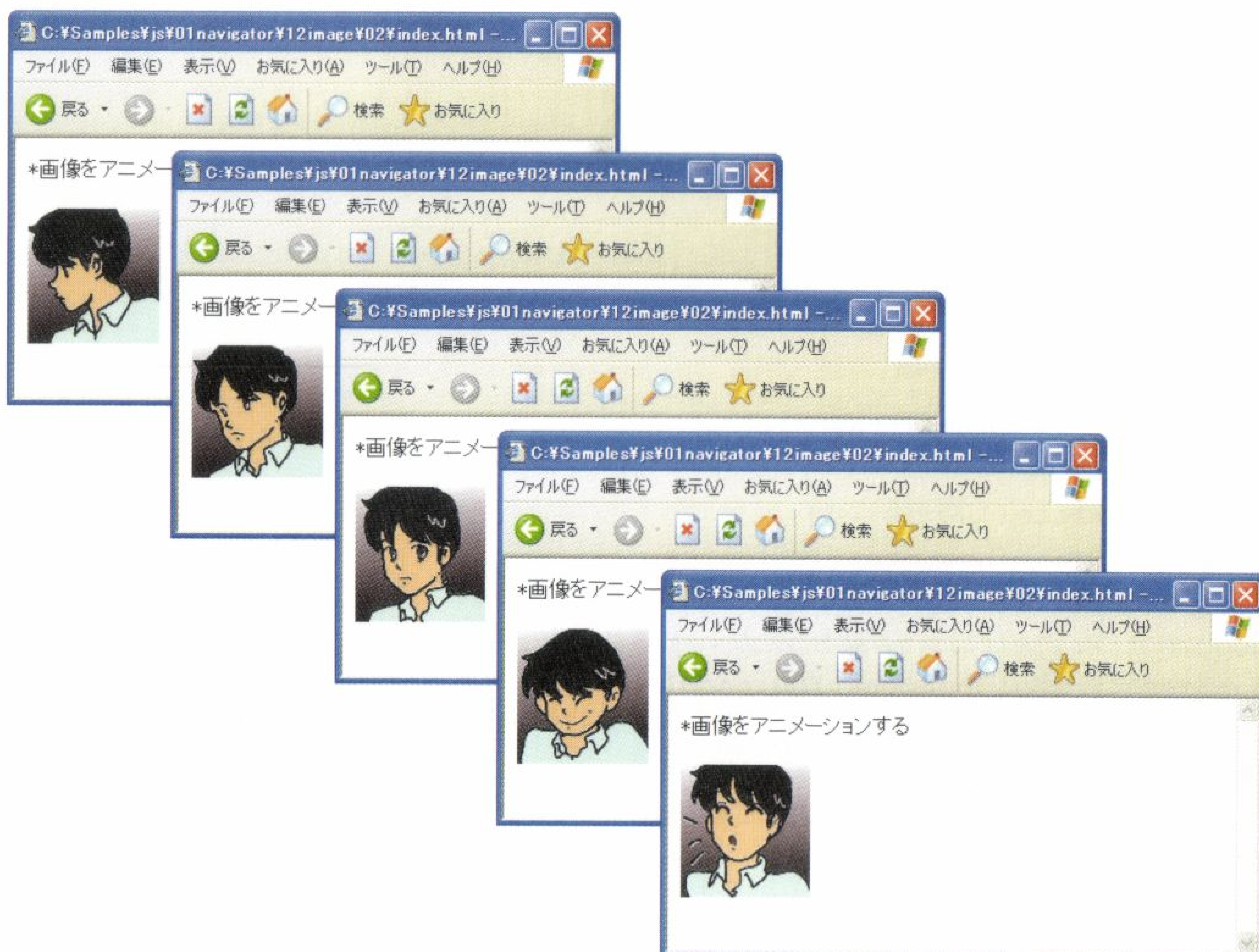
**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
  ～中略～
</head>
<body>
＊画像の情報を取得する
<p>
<img src="image2.jpg" name="IMG" alt="image.jpg" width="100" heigh
t="100" lowsrc="image1.jpg" border="2" hspace="2" vspace="2">
</p>
<p>
<script type="text/javascript">
<!--
document.write("ボーダー：");
document.write(document.IMG.border);
document.write("<br>");
document.write("ロードが終わったか：");
document.write(document.IMG.complete);
document.write("<br>");
document.write("イメージの高さ：");
document.write(document.IMG.height);
document.write("<br>");
document.write("イメージのhspace：");
document.write(document.IMG.hspace);
document.write("<br>");
document.write("lowsrcのURL：");
document.write(document.IMG.lowsrc);
document.write("<br>");
document.write("イメージのURL：");
document.write(document.images[0].src);
document.write("<br>");
document.write("イメージのvspace：");
document.write(document.images[0].vspace);
document.write("<br>");
document.write("イメージの幅：");
document.write(document.images[0].width);
//-->
</script>
</p>
</body>
</html>
```

 <img>：「画像とマルチメディア」の「画像を配置する」(P.99)

# 画像をアニメーションする

オブジェクト名 = new Image()
document.オブジェクト名.src [プロパティ]




「src」プロパティの値を後から変更できることを利用して、複数の画像をリアルタイムに切り替えて、アニメーションのような効果を出すことができます。
サンプルでは、まずimage1.jpgからimage5.jpgまでの5枚の画像を用意し、Arrayオブジェクトを使ってImageオブジェクトの配列を作成しています。配列内の要素には、「ANIMA[1]」から「ANIMA[5]」の5つのImageオブジェクトがあり、それぞれ「ANIMA[1].src」に「image1.jpg」の値、「ANIMA[2].src」に「image2.jpg」の値、といった具合に画像ファイルのURLの値を設定しています。
ページが読み込まれた時に、<body>内に設定したイベントハンドラ「onLoad」が、アニメーションの処理を設定した関数「anime_1()」を発生します。関数の処理では、「document.animation.src」に「ANIMA[1].src」から「ANIMA[5].src」の値を設定し、「setTimeout()」の処理で再び関数を発生しています。「document.animation.src」に設定される値は「ANIMA[1].src」から始まり、「ANIMA[5].src」までくると、13行目から15行目の処理で再び「ANIMA[1].src」に戻ります。これにより、アニメーションの処理は終ることなく繰り返されます。
JavaScript1.1で追加されたオブジェクトです。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
var ImageSetB = 1;
ANIMA = new Array();
for(i = 1; i < 6; i++) {
ANIMA[i] = new Image() ;
ANIMA[i].src = "image" + i + ".jpg" ;
}
function anime_1() {
document.animation.src = ANIMA[ImageSetB].src;
ImageSetB++;
if(ImageSetB > 5) {
ImageSetB = 1;
}
setTimeout("anime_1()",500);
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body onLoad="anime_1()">
*画像をアニメーションする
<p>
<img src="image1.jpg" name="animation" alt="Animation" border="0" width="100" height="100">
</p>
</body>
</html>
```

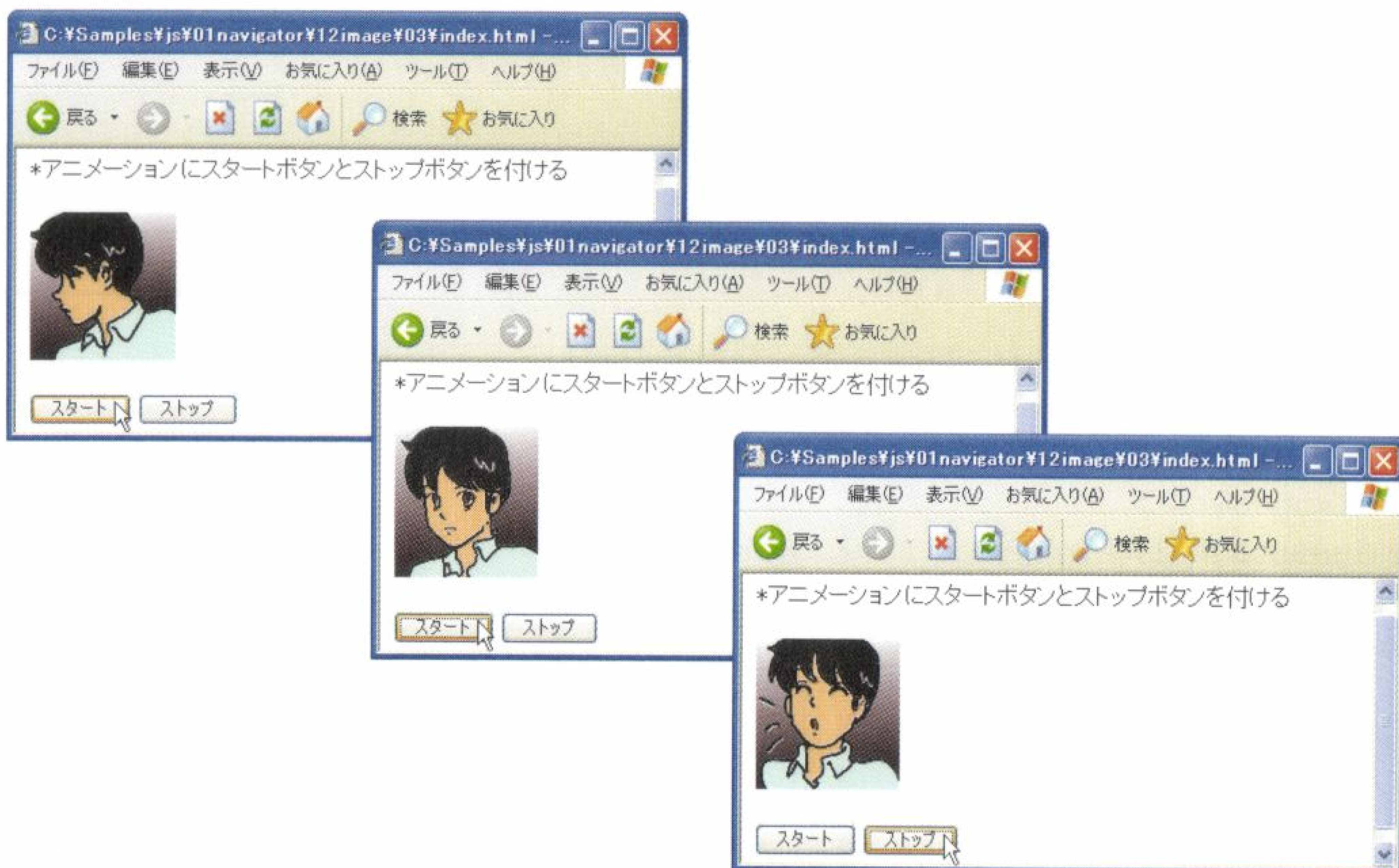

setTimeout():「windowオブジェクト」の「ステータス行に文字を流す」(P.364)
オブジェクト = new Array():「Arrayオブジェクト」の「曜日を表示する - Arrayオブジェクトを使う -」(P.562)

JavaScript



# アニメーションにスタートボタンとストップボタンを付ける

オブジェクト名 = new Image()
document.オブジェクト名.src ［プロパティ］




サンプルでは、前項の「画像をアニメーションする」で作成したImageオブジェクトの配列を利用しています。
［スタート］ボタンが押された時は、関数「anime_2()」が発生してImageオブジェクトの配列の要素が呼び出され、アニメーションがスタートします。［ストップ］ボタンが押された時には、関数「stop()」が発生して、「clearTimeout()」メソッドによってアニメーションが停止します。
「clearTimeout()」メソッドの設定は、「ID名=setTimeout()」と「setTimeout()」メソッドにIDを設定し、そのIDを「clearTimeout(ID名)」と「clearTimeout()」メソッド内で指定することにより行います。
JavaScript1.1で追加されたオブジェクトです。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
```

```
<title></title>

<script type="text/javascript">
<!--
var TimeSet1 = 500;
var ImageSetA = 1;
ANIMA = new Array();
for(i = 1; i < 6; i++) {
   ANIMA[i] = new Image();
   ANIMA[i].src = "image" + i + ".jpg";
}
function anime_2() {
   document.animation.src = ANIMA[ImageSetA].src;
   ImageSetA++;
       if( ImageSetA > 5) {
       ImageSetA = 1;
       }
   timerID=setTimeout("anime_2()", TimeSet1);
}
function stop(){
   clearTimeout(timerID);
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

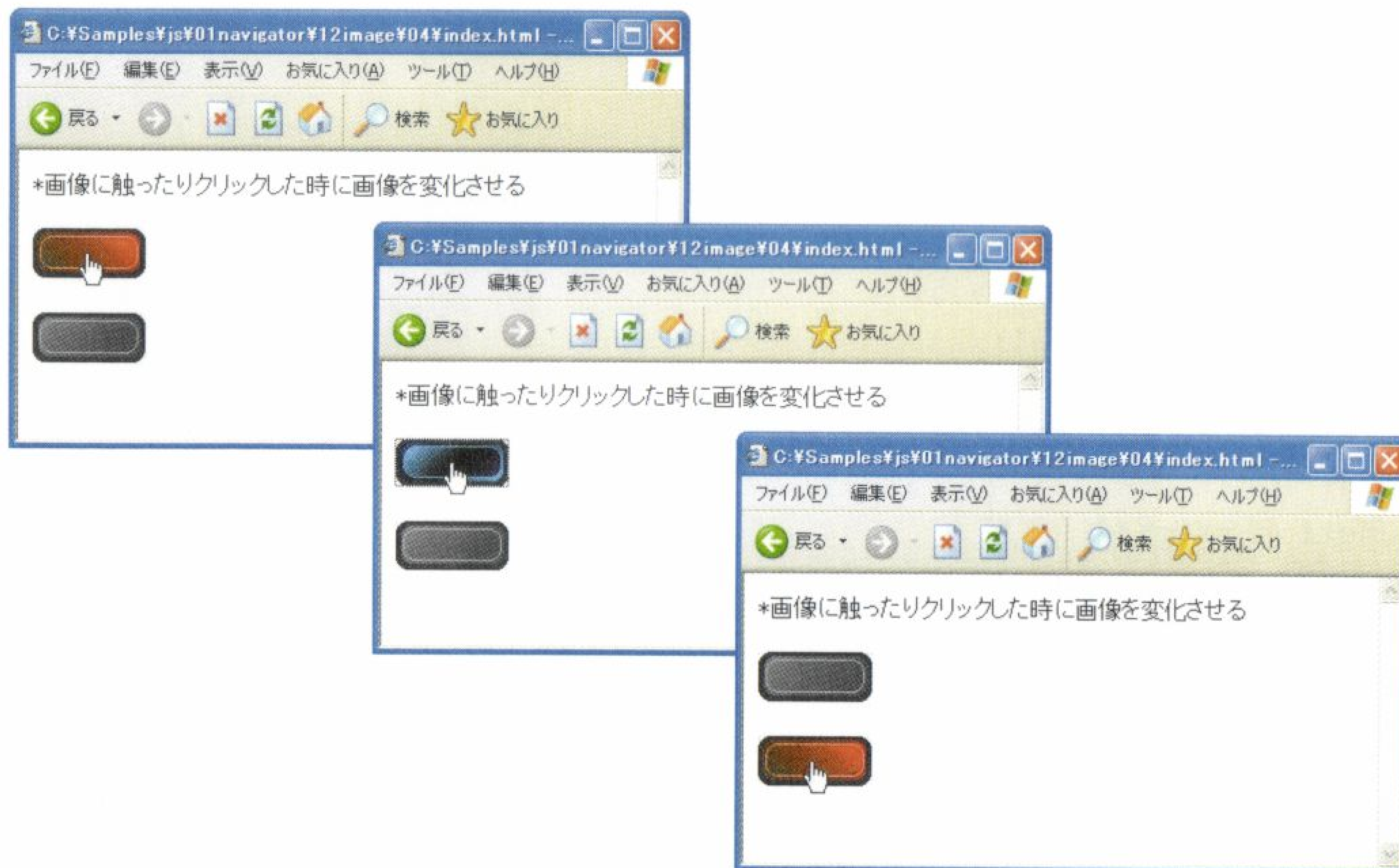
</head>
<body>
*アニメーションにスタートボタンとストップボタンを付ける
<p>
<img src="image1.jpg" name="animation" alt="Animation" border="0"
width="100" height="100">
</p>
<form>
    <input type="button" value=" スタート " onClick="anime_2()">
    <input type="button" value=" ストップ " onClick="stop()">
</form>
</body>
</html>
```


setTimeout()・clearTimeout():「windowオブジェクト」の「ステータス行に文字を流す」(P.364)
オブジェクト = new Array():「Arrayオブジェクト」の「曜日を表示する - Arrayオブジェクトを使う -」(P.562)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Safari IE5-mac IE4-mac

# 画像に触ったりクリックした時に画像を変化させる

オブジェクト名 = new Image()
document.オブジェクト名.src [プロパティ]
document.images[インデックス] [配列]




サンプルではまず、普通の時のボタンの画像「button1.jpg」、マウスが画像の上に乗った時の画像「button2.jpg」、ボタンがクリックされた時の画像「button3.jpg」、の3枚の画像を用意します。そして、「画像をアニメーションする」(P.456)と同じ要領で、それぞれのURLの値を持った3つの配列の要素を作っています。

実際に画像を変化させるのは、リンクの中に設定したイベントハンドラで行います。どの画像を変化させるのかは、HTMLファイルが読み込まれるのと同時に「document.images[0]」から始まるイメージ配列ができているので、それで指定するようにしています。

上の画像上にマウスが乗った時は、イベントハンドラ「onMouseOver」が関数「SetImage1(2,0)」を発生して、「document.images[0]」の位置の画像を「button2.jpg」に置き換えます。マウスをクリックした時は、「onClick="SetImage1(3,0)"」によって、「document.images[0]」の位置のイメージを「button3.jpg」に置き換えます。マウスが画像から離れた時は、「onMouseOut="SetImage1(1,0)"」によって、「document.images[0]」の位置のイメージを「button1.jpg」に置き換えています。

また、下の画像上にマウスが乗ったり、クリックされたり、マウスが離れた時は、「document.images[1]」の位置の画像が置き換わるように設定しています。

JavaScript1.1で追加されたオブジェクトです。

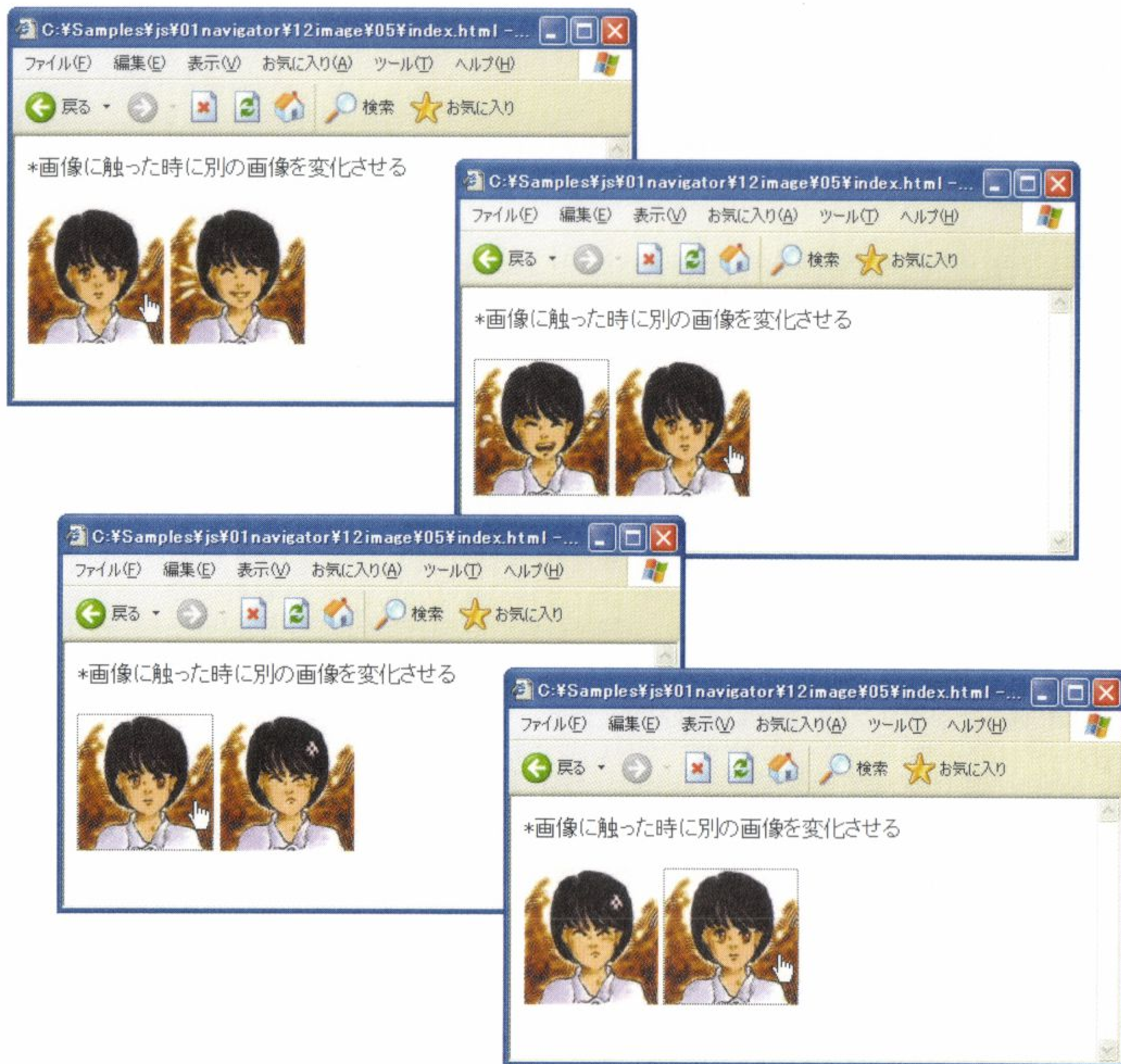
## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
var ButtonImage = new Array();
    for(i = 1; i < 4; i++) {
        ButtonImage[i]= new Image();
        ButtonImage[i].src="button" + i + ".jpg";
    }
function SetImage1(flag, position) {
        document.images[position].src=ButtonImage[flag].src;
}
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*画像に触ったりクリックした時に画像を変化させる
<p>
<a href="#" onMouseOver="SetImage1(2,0)"
onMouseOut="SetImage1(1,0)" onClick="SetImage1(3,0)">
<img src="button1.jpg" alt="button1" border=0 width="78" heigh
t="33"></a>
</p>
<p>
<a href="#" onMouseOver="SetImage1(2,1)" onMouseOut="SetImage1
(1,1)" onClick="SetImage1(3,1)">
<img src="button1.jpg" alt="button2" border=0 width="78" heigh
t="33"></a>
</p>
</body>
</html>
```


オブジェクト = new Array():「Arrayオブジェクト」の「曜日を表示する - Arrayオブジェクトを使う -」(P.562)

# 画像に触った時に別の画像を変化させる

オブジェクト名 = new Image()
document.オブジェクト名.src　［プロパティ］
document.images[インデックス]　［配列］




「画像に触ったりクリックした時に画像を変化させる」(P.460)と同じ要領で、画像に触ったタイミングで別の画像を置き換えることができます。
サンプルでは、イメージ配列「document.images[0]」の画像の上にマウスが乗ったり、クリックされたり、マウスが離れた時に、「document.images[1]」の画像が置き換わるようにしています。
JavaScript1.1で追加されたオブジェクトです。

## Sample

```
<script type="text/javascript">
<!--
OnMouse = new Array();
    for(i = 1; i < 5; i++) {
        OnMouse[i] = new Image();
        OnMouse[i].src = "image" + i + ".jpg";
    }
function OnMoSet1(flag, position) {
    document.images[position].src=OnMouse[flag].src;
    return false;
}
//-->
</script>
  ～中略～
</head>
<body>
＊画像に触った時に別の画像を変化させる
<p>
<a href="#" onMouseOver="OnMoSet1(2,1)" onMouseOut="OnMoSet1(1,1)" 
onClick="return OnMoSet1(4,1)">
<img src="image1.jpg" alt="onmousea-1" border=0 width="100" height 
="100"></a>
<a href="#" onMouseOver="OnMoSet1(3,0)" onMouseOut="OnMoSet1(1,0)" 
onClick="return OnMoSet1(4,0)">
<img src="image1.jpg" alt="onmousea-2" border=0 width="100" height 
="100"></a>
</p>
</body>
```

オブジェクト = new Array():「Arrayオブジェクト」の「曜日を表示する - Arrayオブジェクトを使う -」(P.562)

### 注意

**マウス操作のタイミングで画像を置き換える時の注意**

Imageオブジェクトでは、イベントハンドラの「onClick」「onMouseOver」「onMouseOut」はサポートされていません。

したがって、マウスが画像上に乗った時や離れた時などに画像の置き換えなどの処理を行う場合は、このサンプルのように、Linkオブジェクト内などにイベントハンドラを置く必要があります。

もし、画像をクリックした時に違うページへ飛びたくない場合は、このサンプルのように、「return false」を返してイベントの処理を中止させるか、「href="javascript:"」を使えば大丈夫です。

JavaScript

# 別フレームの画像を変化させる

オブジェクト名 = new Image()
document.オブジェクト名.src [プロパティ]
document.images[インデックス] [配列]




「画像に触ったりクリックした時に画像を変化させる」(P.464)と同じ要領で、別フレームの画像を置き換えることができます。その場合は、置き換える画像の指定は、「parent.画像のあるフレーム名.document.images[インデックス].src」とします。
JavaScript1.1で追加されたオブジェクトです。
実際に試す場合には、この他にも「f3.html」「fp1.html」〜「fp3.html」の4つのHTMLファイルを用意してください。

## Sample

【フレームウィンドウ】

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" >
<html>
<head>
<title></title>
</head>
<frameset cols="140,*">
    <frameset rows="140,*">
        <frame src="f1.html" name="f1">
        <frame src="f2.html" name="f2">
    </frameset>
<frame src="f3.html" name="f3">
```

```
</frameset>
<noframes>
フレーム機能を使用しています。フレーム対応のブラウザで試してください(^_^)。
</noframes>
</html>
```

【f1.html】

```
<body>
<img src="image1.jpg" alt="OnMouseB-1" width="100" height="100">
</body>
```

【f2.html】

```
<script type="text/javascript">
<!--
OnMouseB = new Array();
for(i = 1; i < 6; i++) {
   OnMouseB[i] = new Image();
   OnMouseB[i].src = "image" + i + ".jpg";
}
function OnMoSet2(flag, position) {
    parent.f1.document.images[position].src=OnMouseB[flag].src;
}
//-->
</script>
  ～中略～
<body>
<p>
・<a href="fp1.html" target="f3" onMouseOver="OnMoSet2(2,0)" onMou
seOut="OnMoSet2(1,0)">笑顔</a>
</p>
<p>
・<a href="fp2.html" target="f3" onMouseOver="OnMoSet2(3,0)" onMou
seOut="OnMoSet2(1,0)">泣き顔</a>
</p>
<p>
・<a href="fp3.html" target="f3" onMouseOver="OnMoSet2(4,0)" onMou
seOut="OnMoSet2(1,0)">怒り顔</a>
</p>
<p>
・<a href="f3.html" target="f3" onMouseOver="OnMoSet2(5,0)" onMous
eOut="OnMoSet2(1,0)">始めへ戻る</a>
</p>
</body>
```

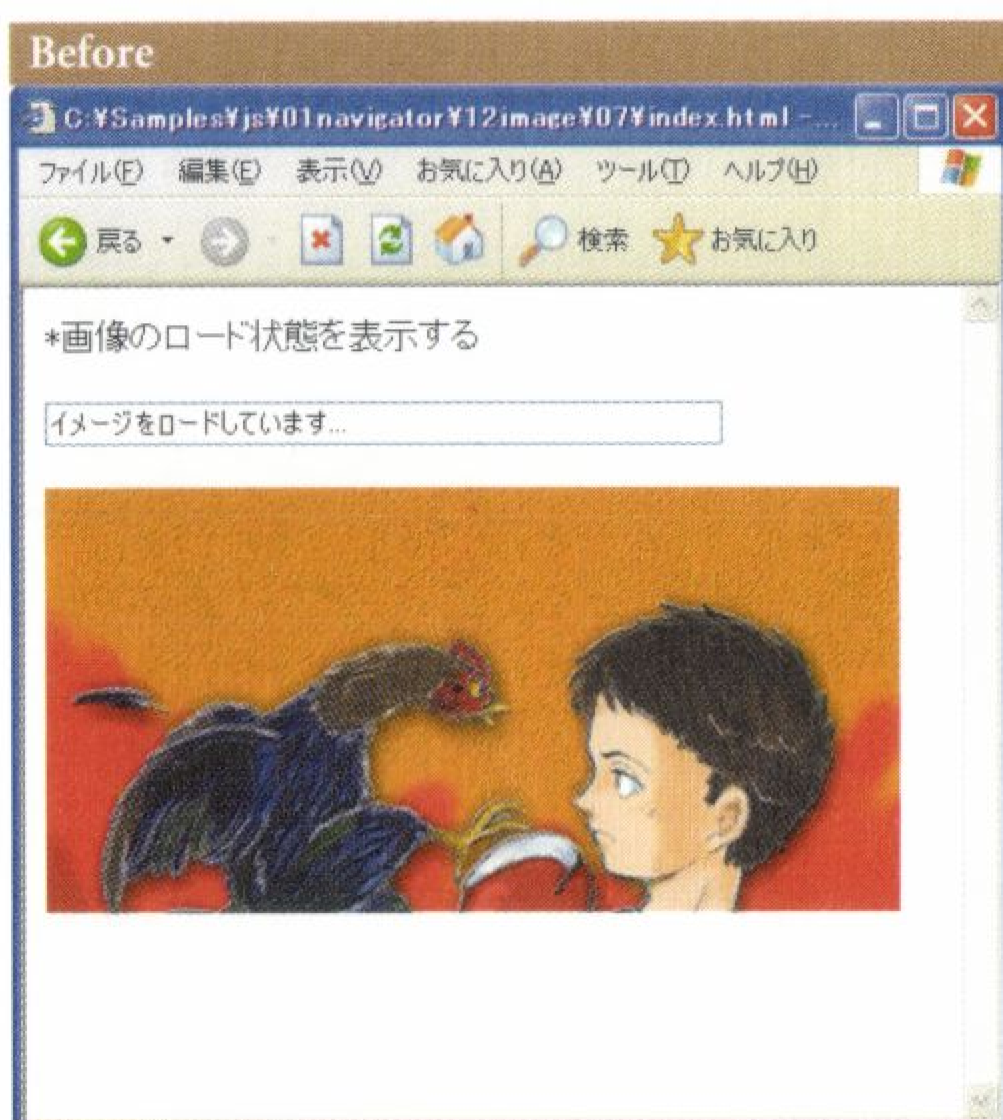

parent.フレーム名:「fram オブジェクト」の「入力されたURLを別フレームに表示する」(P.374)
オブジェクト = new Array():「Arrayオブジェクト」の「曜日を表示する - Arrayオブジェクトを使う -」(P.562)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

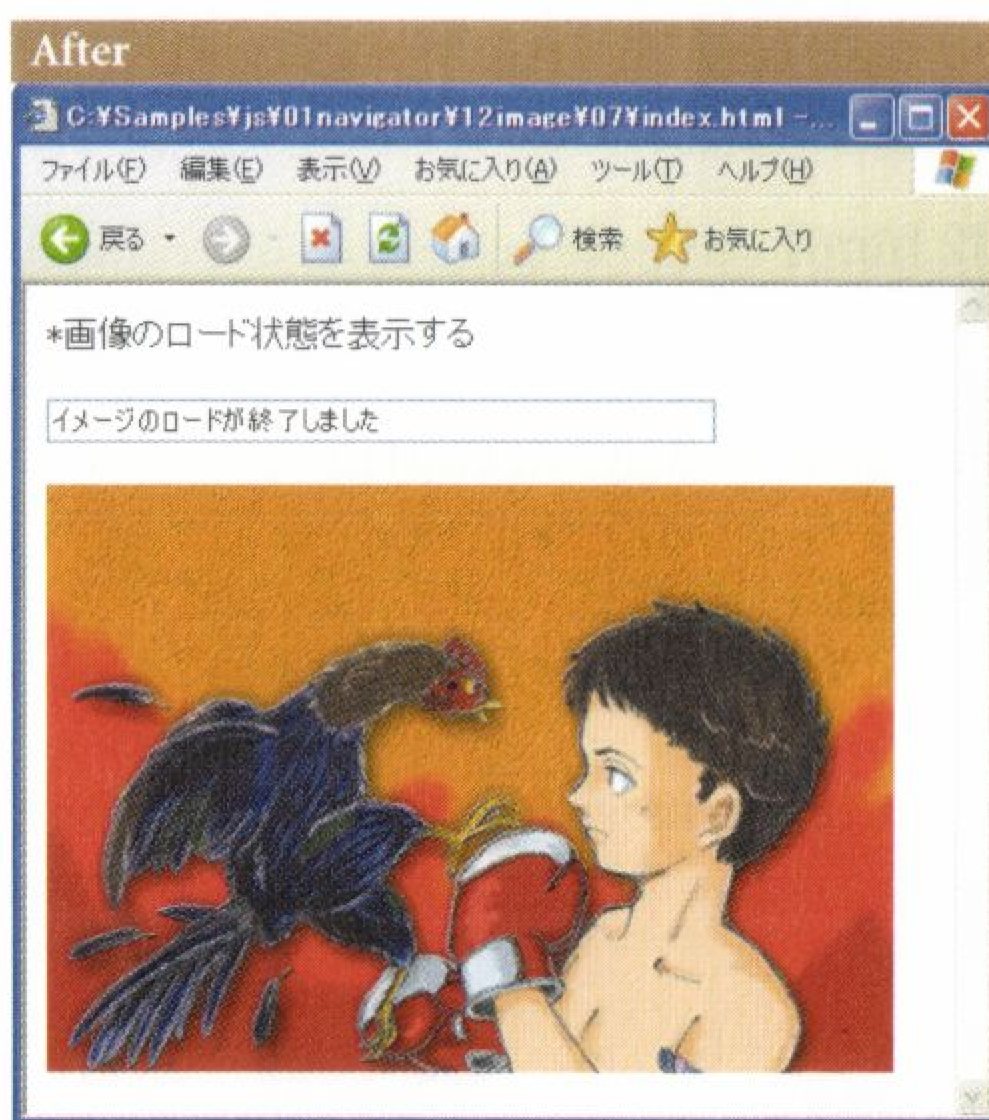
# 画像のロード状態を表示する

onAbort="スクリプト | 関数" [イベントハンドラ]
onError="スクリプト | 関数" [イベントハンドラ]
onLoad="スクリプト | 関数" [イベントハンドラ]

Before




After




イベントハンドラ「onAbort」は、画像読み込み中に[中止(Stop)]ボタンが押されるなどで画像の読み込みが中止された時に、イベントハンドラ「onError」は画像読み込みエラー時に、イベントハンドラ「onLoad」は画像読み込み終了時に、それぞれイベントを発生します。
サンプルでは、画像ファイルのそれぞれの状態を取得し、それに合わせたメッセージをフォーム内に書き出しています。
このサンプルで、画像の後にフォームを持って来るようにすると、スクリプトが機能しない場合があります。
JavaScript1.1で追加されたオブジェクトです。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function STOP(){ document.ZYOUTAI.zyo.value = "イメージのロードが中止さ
れました" }
function ERR(){ document.ZYOUTAI.zyo.value = "イメージのロードに失敗しま
した" }
```

```
function OK(){ document.ZYOUTAI.zyo.value = "イメージのロードが終了しました" }
//-->
</script>
  ～中略～
</head>
<body>
*画像のロード状態を表示する
<p>
<form name="ZYOUTAI">
<input type="text" name="zyo" value="イメージをロードしています..." size="60">
</form>
</p>
<p>
<img src="image.jpg" alt="image.jpg" width="400" height="278" onAbort="STOP()" onError="ERR()" onLoad="OK()">
</p>
</body></html>
```


document.フォームオブジェクト名.オブジェクト名.value：「formオブジェクト」の「メール送信時に挨拶を表示する」(P.437)
onAbort：リファレンス「イベントハンドラ」の「onAbort」(P.619)
onError：リファレンス「イベントハンドラ」の「onError」(P.620)
onLoad：リファレンス「イベントハンドラ」の「onLoad」(P.620)

**注意**

## 大きな画像やフォーム、テーブルとJavaScriptを共存させる時の注意

大きな画像やフォーム、テーブルなど、ブラウザがレイアウトを完成するのに時間がかかるようなHTMLの記述の後にJavaScriptを記述しても、スクリプトがうまく動かない場合があります。

これは、ブラウザのレイアウト確定に邪魔され、JavaScriptの認識が行われなかったために起こる現象のようです。

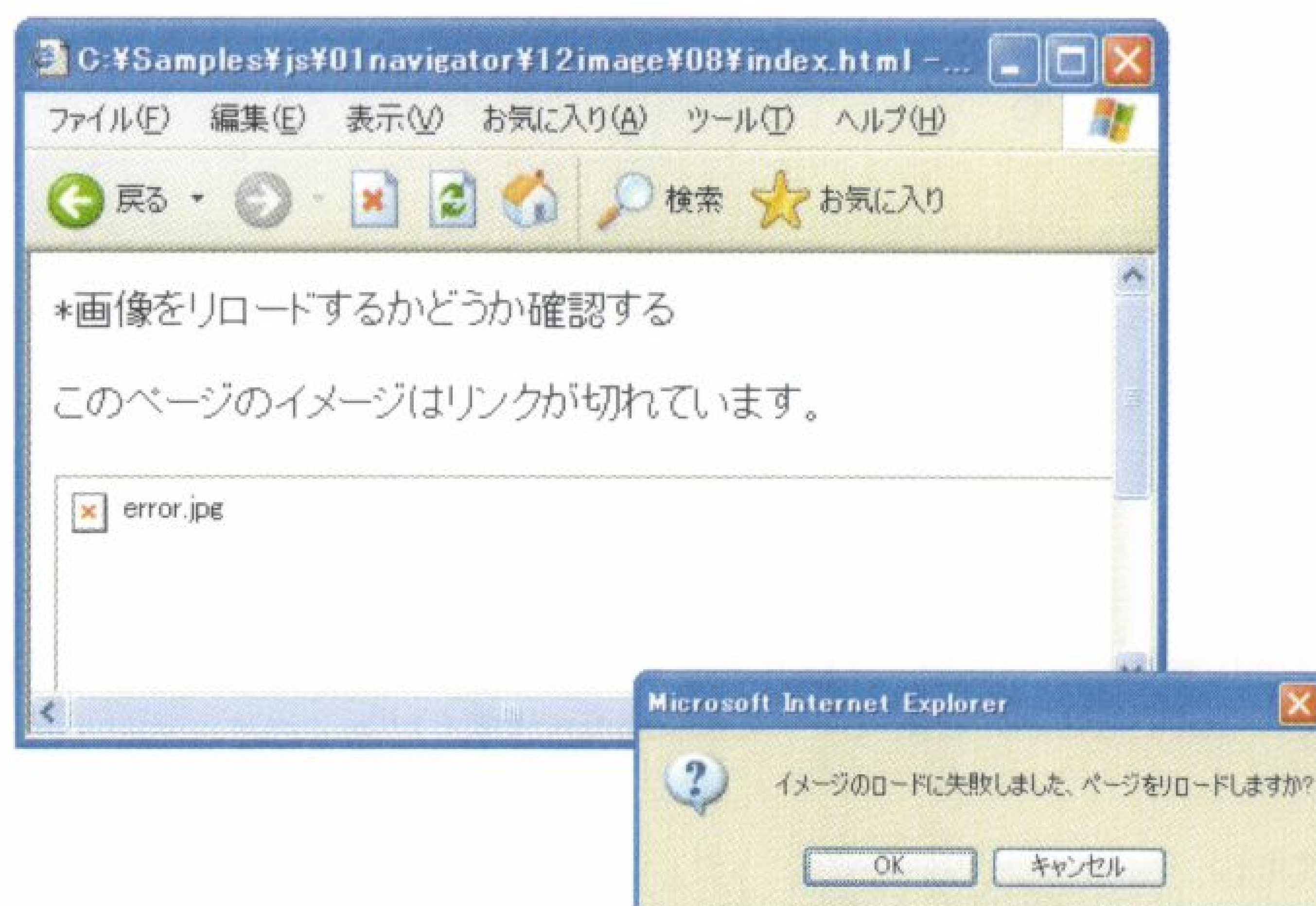
画像の場合は、Netscape Navigator 3.0以降のブラウザを使ったり、そうでなくても「width」属性と「height」属性の指定をすれば、画像部分のレイアウト確定が早く行われるので、この問題はほぼ解消されます。しかし、テーブル内にJavaScriptを記述した時や、このサンプルのように大きな画像の上ではなく、下にフォームと組み合わせたJavaScriptを組み込んだ時などに、問題が起こる場合が多いようです。

これらの問題はNetscape Navigator 3.0以降のブラウザでも発生する場合がありますが、Internet Explorerでこのような現象を確認したことはありません。

IE6.0
IE5.5
IE5.0
IE4.0
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

# 画像をリロードするかどうか確認する

**onError="スクリプト | 関数"** [イベントハンドラ]




サンプルでは、画像の読み込みエラー時にイベントを発生するイベントハンドラ「onError」を利用して、画像が正常に読み込まれなかった時に、ページをリロードするかどうかを確認するダイアログボックスを開いています。
JavaScript1.1で追加されたオブジェクトです。

## Sample

```
<script type="text/javascript">
<!--
function ERR2(){
    if ( confirm ("イメージのロードに失敗しました。ページをリロードしますか?") )
{ location.href="08ima.html" }
}
//-->
</script>
  ～中略～
<body>
*画像をリロードするかどうか確認する
<p>このページのイメージはリンクが切れています。</p>
<p><img src="error.jpg" alt="error.jpg" width="474" height="198" on
Error="ERR2()"></p>
</body>
```

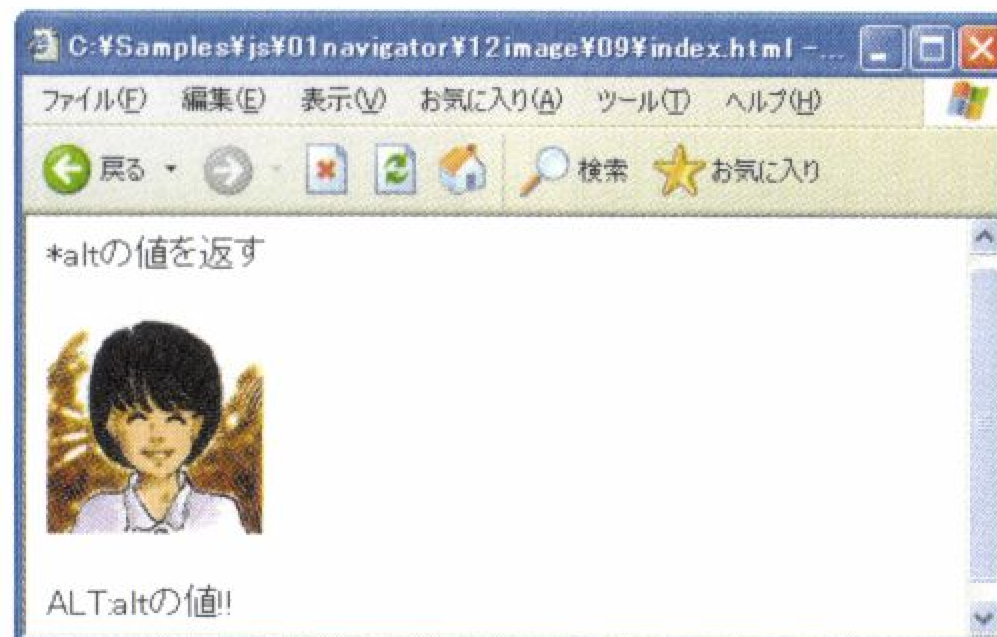

confirm():「windowオブジェクト」の「確認ボタン付きのダイアログボックスを開く」(P.334)
location.href:「locationオブジェクト」の「自ページのURLを取得する」(P.408)
onError:リファレンス「イベントハンドラ」の「onError」(P.620)

イメージを操作する

# altの値を返す

**document.getElementById(オブジェクト名).alt** [プロパティ]
**document.all(オブジェクト名).alt** [プロパティ]






DOMの採用により、Internet Explorer 4.0とNetscape 6.0から、基本的にHTMLタグのあらゆる属性へ、JavaScriptからアクセスできるようになりました。<img>の「alt」属性も、その中のひとつです。
サンプルでは、Internet Explorerでは「all()」メソッドを使い、Netscapeでは「getElementById()」メソッドを使い、「alt」の値を取得しています。

## Sample

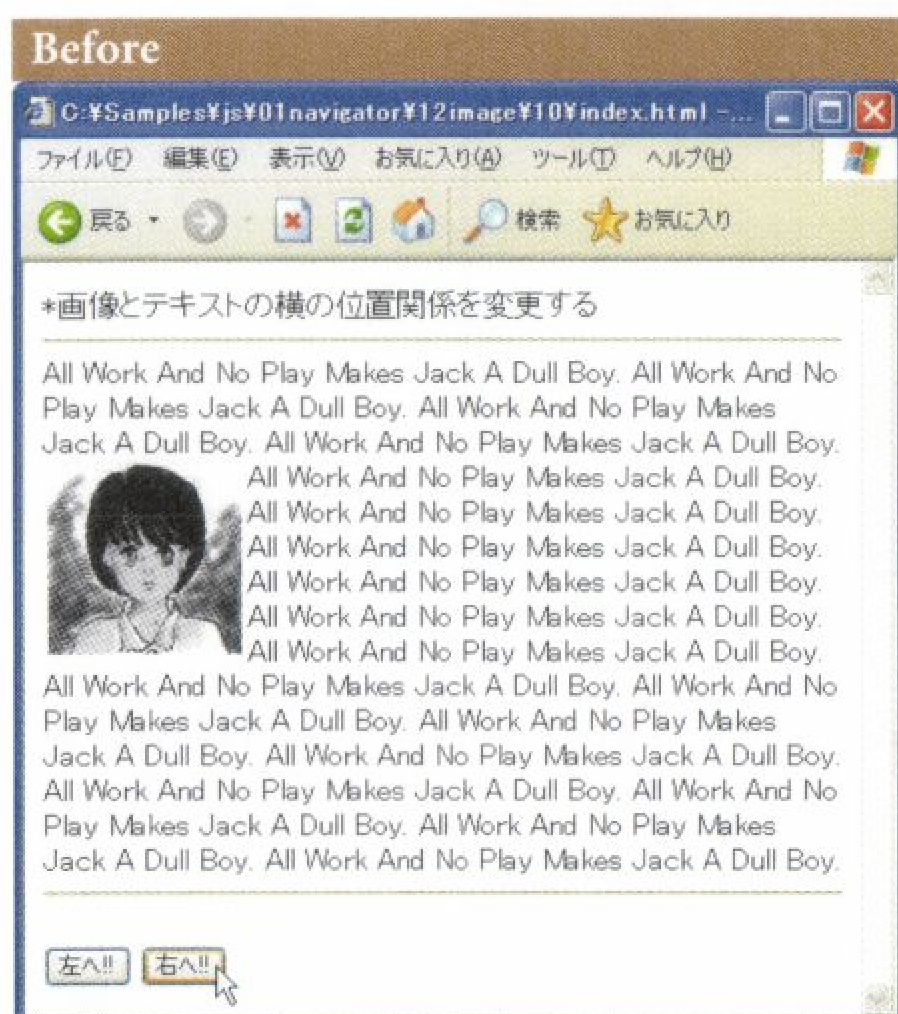
```
<body>
*altの値を返す
<p>
<img src="image.jpg" id="IMG" width="100" height="100" alt="altの値!!">
</p>
<p>
<script type="text/javascript">
<!--
document.write("ALT:");
if( navigator.appName.charAt(0)=="N" ){
    if(document.getElementById){document.write(document.getElementBy
Id("IMG").alt)}
}
if( navigator.appName.charAt(0)=="M" ){
        if(document.all){document.write(document.all("IMG").alt)}
}
//-->
</script>
</p>
</body>
```


<img alt="">：「画像とマルチメディア」の「画像を配置する」(P.99)
付録コラム「DynamicHTMLとは」(P.652)

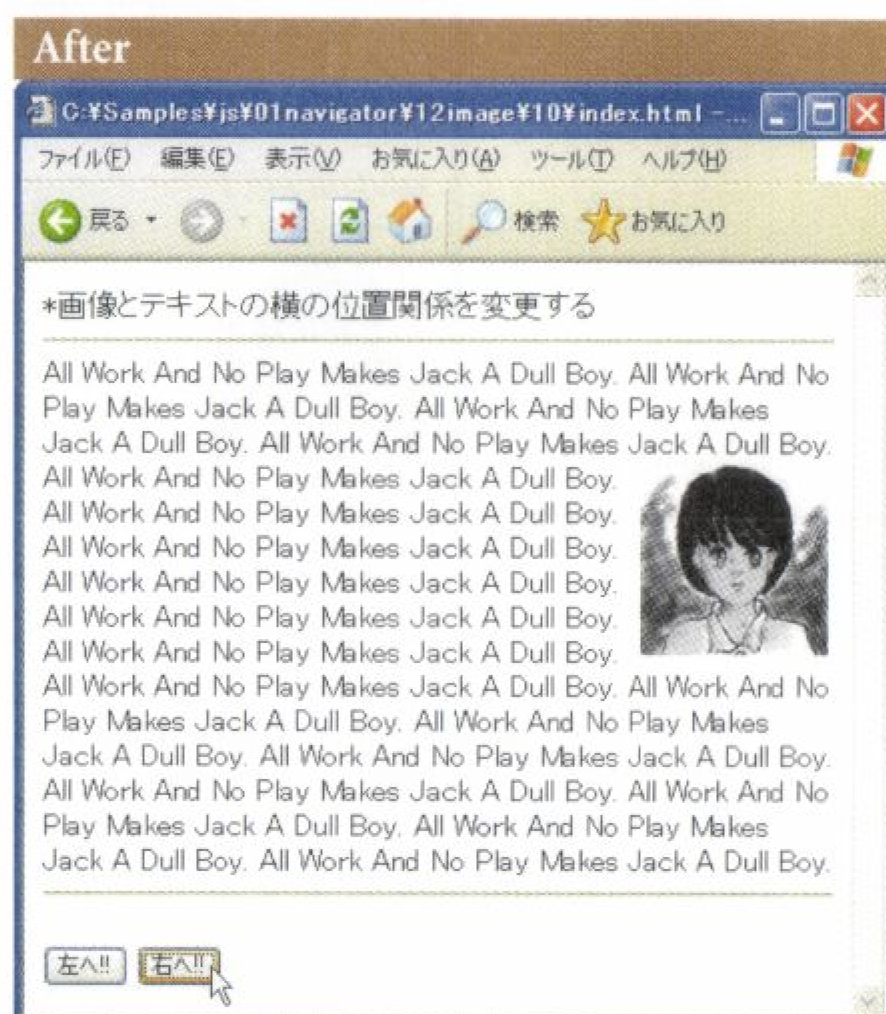
# 画像とテキストの横の位置関係を変更する

**document.getElementById(*オブジェクト名*). align="*left* | *right*"** [プロパティ]
**document.all(*オブジェクト名*).align="*left* | *right*"** [プロパティ]

Before



After



サンプルでは、DOMの採用により<img>の「align」属性が操作できるようになったことを利用して、画像に対するテキストの横の位置関係を変更しています。
なお、サンプルでは、Internet Explorerの4.Xとそれより後のバージョンでのDOMの実装の違いを考慮して、「getElementById()」メソッドと「all()」メソッドのどちらをサポートしたブラウザでも、対応できるようにしています。
[右へ!!]ボタンをクリックすると関数「AlignRight()」が発生し、「document.getElementById("IMG").align="right"」か「document.all("IMG").align="left"」が評価され、「align」に「right」の値が設定され、画像が右へ移動します。同様に、[左へ!!]ボタンをクリックした時は関数「AlignLeft()」が発生し、「align」に「left」の値が設定されるので、画像は右へ移動します。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function AlignLeft(){
```

```
    if(document.getElementById){document.getElementById("IMG").ali
gn="left"}
    if(document.all){document.all("IMG").align="left"}
}
function AlignRight(){
    if(document.getElementById){document.getElementById("IMG").ali
gn="right"}
    if(document.all){document.all("IMG").align="right"}
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*画像とテキストの横の位置関係を変更する
<hr>
All Work And No Play Makes Jack A Dull Boy. All Work And No Play
Makes Jack A Dull Boy. All Work And No Play Makes Jack A Dull Boy.
All Work And No Play Makes Jack A Dull Boy.  <img id="IMG"
align="left" src="image.jpg" width="100" height="100"
alt="align">All Work And No Play Makes Jack A Dull Boy. All Work
And No Play Makes Jack A Dull Boy. All Work And No Play Makes Jack
A Dull Boy. All Work And No Play Makes Jack A Dull Boy. All Work
And No Play Makes Jack A Dull Boy. All Work And No Play Makes Jack
A Dull Boy. All Work And No Play Makes Jack A Dull Boy. All Work
And No Play Makes Jack A Dull Boy. All Work And No Play Makes Jack
A Dull Boy. All Work And No Play Makes Jack A Dull Boy. All Work
And No Play Makes Jack A Dull Boy. All Work And No Play Makes Jack
A Dull Boy. All Work And No Play Makes Jack A Dull Boy. All Work
And No Play Makes Jack A Dull Boy.
<hr>
<form name="FORM">
    <input type="button" value="左へ!!" onClick="AlignLeft()">
    <input type="button" value="右へ!!" onClick="AlignRight()">
</form>
</body>
</html>
```


<img align="">：「画像とマルチメディア」の「画像にテキストを回り込ませる」(P.105)
付録コラム「DynamicHTMLとは」(P.652)

# 画像とテキストの縦の位置関係を変更する

**document.getElementById(*オブジェクト名*). align="*top* | *middle* | *bottom*"** [プロパティ]
**document.all(*オブジェクト名*).align="*left* | *right*"** [プロパティ]




DOMの採用により、ブラウザのより細かい部分を、動的に変更することができるようになりました。
サンプルでは、ボタンクリックのタイミングで「align」にそれぞれの値を設定することにより、画像に対するテキストの縦の位置関係を動的に変更しています。
「top」の値を設定した時は画像の上とテキストの上を、「middle」の値を設定した時は画像の中心とテキストのベースラインを、「bottom」の値を設定した時は画像の下とテキストのベースラインを、それぞれ揃えます。
なお、サンプルでは、Internet Explorerの4.Xとそれより後のバージョンでのDOMの実装の違いを考慮して、「getElementById()」メソッドと「all()」メソッドのどちらをサポートしたブラウザでも、対応できるようにしています。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
```

```
<title></title>

<script type="text/javascript">
<!--
function AlignTop(){
    if(document.getElementById){document.getElementById("IMG").ali
gn="top"}
    if(document.all){document.all("IMG").align="top"}
}
function AlignMiddli(){
    if(document.getElementById){document.getElementById("IMG").ali
gn="middle"}
    if(document.all){document.all("IMG").align="middle"}
}
function AlignBottom(){
    if(document.getElementById){document.getElementById("IMG").ali
gn="bottom"}
    if(document.all){document.all("IMG").align="bottom"}
}
//-->
</script>

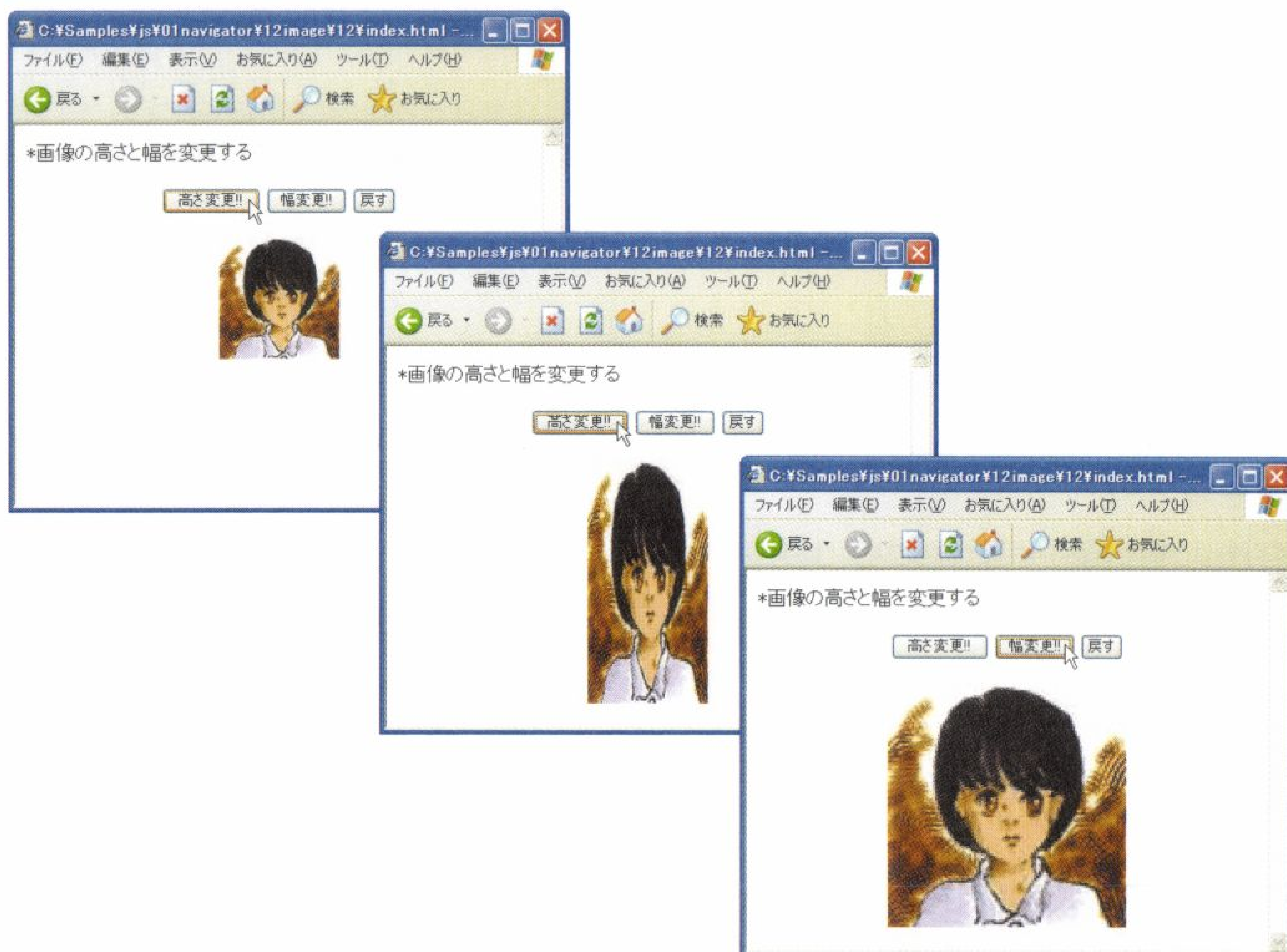
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*画像とテキストの縦の位置関係を変更する
<p>
All Work <img id="IMG" align="middle" src="image.jpg" width="100"
height="100" alt="align"> And No Play Makes Jack A Dull Boy.
</p>
<p>
<form name="FORM">
    <input type="button" value="AlignTop" onClick="AlignTop()">
    <input type="button" value="AligMniddle" onClick="AlignMiddli
()">
    <input type="button" value="AlignBottom" onClick="AlignBottom
()">
</form>
</p>
</body>
</html>
```


<img align="">：「画像とマルチメディア」の「画像にテキストを回り込ませる」(P.105)
付録コラム「DynamicHTMLとは」(P.652)

# 画像の高さと幅を変更する

**document.getElementById(*オブジェクト名*).height=*"ピクセル"*** [プロパティ]
**document.getElementById(*オブジェクト名*).width=*"ピクセル"*** [プロパティ]
**document.all(*オブジェクト名*).height=*"ピクセル"*** [プロパティ]
**document.all(*オブジェクト名*).width=*"ピクセル"*** [プロパティ]




Internet Explorerでは、以前から画像のサイズを動的に変更することが可能でしたが、Netscape Navigatorでも、DOMが採用されたNetscape 6.0から可能になりました。
サンプルでは、ボタンクリックのタイミングで関数を発生させ、関数内の処理で「height」プロパティや「width」プロパティの値に数値を設定することによって、画像の高さや幅を動的に変更しています。[高さ変更!!]ボタンをクリックすると画像の高さが200ピクセルに、[幅変更!!]ボタンをクリックすると画像の横幅が200ピクセルになります。また、[戻す]ボタンをクリックすると、画像の高さと幅が、それぞれの初期状態の100ピクセルになります。
なお、サンプルでは、Internet Explorerの4.Xとそれより後のバージョンでのDOMの実装の違いを考慮して、「getElementById()」メソッドと「all()」メソッドのどちらをサポートしたブラウザでも、対応できるようにしています。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function ImgHeight(){
    if(document.getElementById){document.getElementById("IMG").
height = "200"}
    if(document.all){document.all("IMG").height = "200"}
}
function ImgWidth(){
    if(document.getElementById){document.getElementById("IMG").
width = "200"}
    if(document.all){document.all("IMG").width = "200"}
}
function Replace(){
    if(document.getElementById){document.getElementById("IMG").
height = "100";
        document.getElementById("IMG").width = "100";
    }
    if(document.all){document.all("IMG").height = "100";
       document.all("IMG").width = "100";
    }
}
//-->
</script>
  ～中略～
</head>
<body>
*画像の高さと幅を変更する
<center>
<p>
<form>
    <input type="button" value="高さ変更!!" onClick="ImgHeight()">
    <input type="button" value="幅変更!!" onClick="ImgWidth()">
    <input type="button" value="戻す" onClick="Replace()">
</form>
</p>
<img src="image.jpg" id="IMG" alt="image.jpg" width="100" height="100">
</center>
</body>
</html>
```

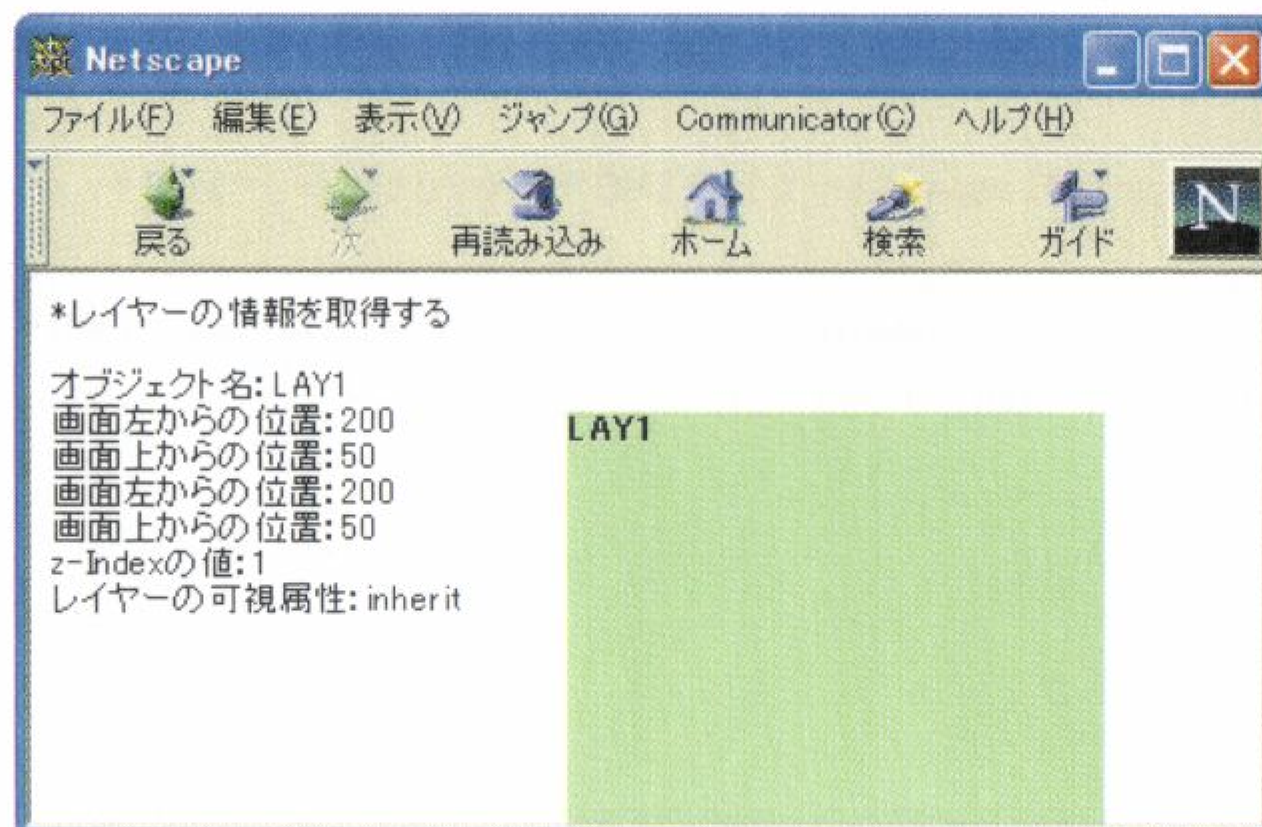
付録コラム「DynamicHTMLとは」(P.652)

# レイヤーの情報を取得する

【オブジェクト】
document.layers[オブジェクト名]
document.layers.オブジェクト名
document.オブジェクト名
document.layers[インデックス]

document.layers[オブジェクト名].name [プロパティ]
document.layers[オブジェクト名].left [プロパティ]
document.layers[オブジェクト名].top [プロパティ]
document.layers[オブジェクト名].pageX [プロパティ]
document.layers[オブジェクト名].pageY [プロパティ]
document.layers[オブジェクト名].zIndex [プロパティ]
document.layers[オブジェクト名].visibility [プロパティ]






Layerオブジェクトはdocumentオブジェクトのプロパティなので、オブジェクトの参照は、サンプルのように「document.layers[オブジェクト名]」「document.layers.オブジェクト名」「document.オブジェクト名」、またはオブジェクトを配列として取り扱って「document.layers[インデックス]」とすることにより可能です。

「name」プロパティは<layer>内の「name」属性で設定したlayerオブジェクト名を、「left」プロパティはページあるいは親レイヤーの左上角からのレイヤー左上角のX軸上の位置の値を、「top」プロパティはページあるいは親レイヤーの左上角からのレイヤー左上角のX軸上の位置の値を、「pageX」プロパティはページ左上角からのレイヤー左上角のX軸上の位置の値を、「pageY」プロパティはページ左上角からのレイヤー左上角のY軸上の位置の値を、「zIndex」プロパティはレイヤーの軸上の値を、「visibility」プロパティはレイヤーの可視属性の値を、それぞれ持っています。

親レイヤーの場合はページ左上角からの位置を参照するので、「left」プロパティと「pageX」プロパティ、「top」プロパティと「pageY」プロパティの値は同じになります。また、サンプルでは、レイヤーに可視属性を設定していないので、親レイヤーの可視属性に準ずる「inherit」の値が返ります。
これらのプロパティは変更可能なので、ページ表示後にこれらの値を変更することにより、レイヤーを移動したり、可視属性を変化することが可能です。
Layerオブジェクトは、Netscape Navigator 4.Xのみでサポートされています。

## Sample

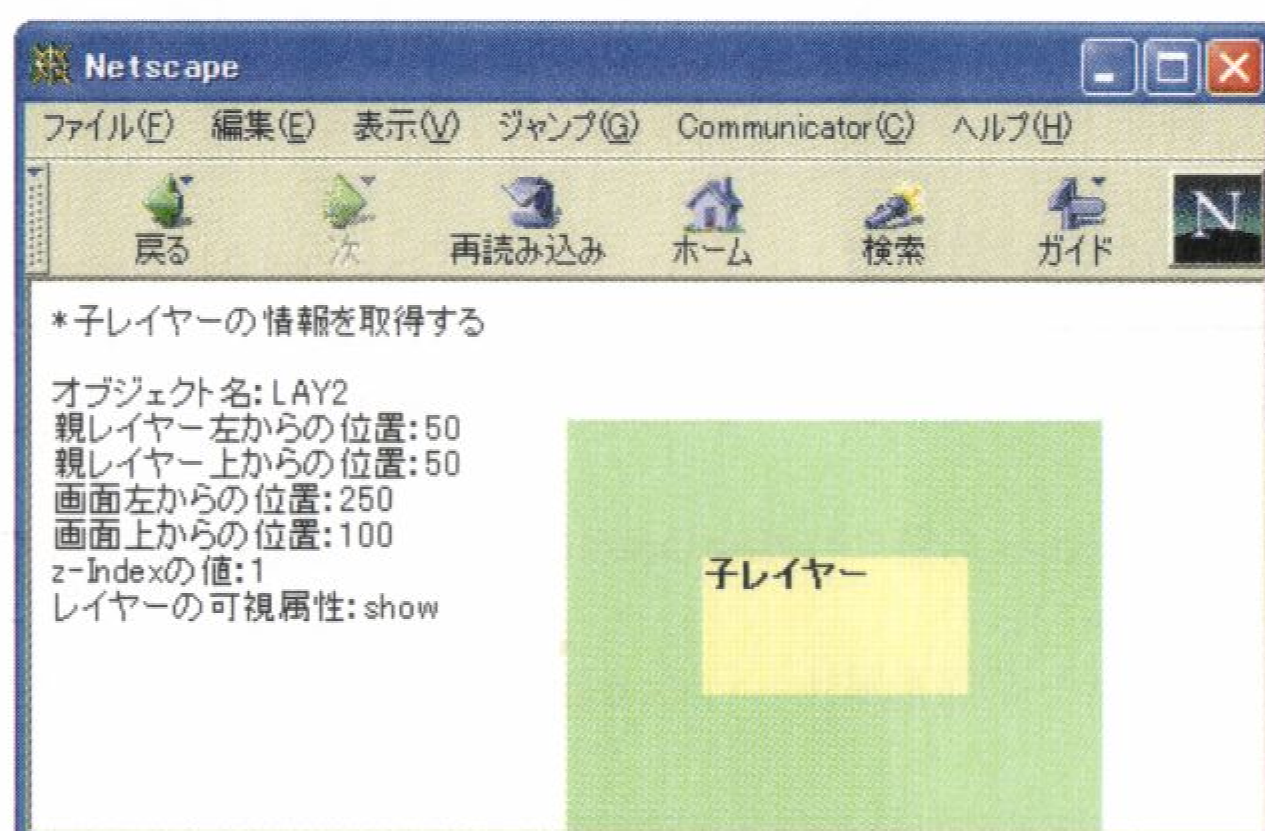
```
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<title></title>
</head>
<body bgcolor="#ffffff">
*レイヤーの情報を取得する
<layer name="LAY1" left="200" top="50" width="200" height="150" z-
index="1" bgcolor="#99ffcc">
<b>LAY1</b>
</layer>
<p>
<script type="text/javascript">
<!--
document.write("オブジェクト名：",document.layers["LAY1"].name);
document.write("<br>");
document.write("画面左からの位置：",document.layers.LAY1.left);
document.write("<br>");
document.write("画面上からの位置：",document.LAY1.top);
document.write("<br>");
document.write("画面左からの位置：",document.layers[0].pageX);
document.write("<br>");
document.write("画面上からの位置：",document.layers["LAY1"].pageY);
document.write("<br>");
document.write("z-Indexの値：",document.layers["LAY1"].zIndex);
document.write("<br>");
document.write("レイヤーの可視属性：",document.layers["LAY1"].visibili
ty);
//-->
</script>
</p>
</body>
</html>
```

# 子レイヤーの情報を取得する

【オブジェクト】
document.layers[オブジェクト名].layers[オブジェクト名]
document.layers.オブジェクト名.layers.オブジェクト名
document.layers[インデックス].layers[インデックス]

document.layers[親レイヤー名].layers[子レイヤー名].name
【プロパティ】
document.layers[親レイヤー名].layers[子レイヤー名].left
【プロパティ】
document.layers[親レイヤー名].layers[子レイヤー名].top
【プロパティ】
document.layers[親レイヤー名].layers[子レイヤー名].pageX
【プロパティ】
document.layers[親レイヤー名].layers[子レイヤー名].pageY
【プロパティ】
document.layers[親レイヤー名].layers[子レイヤー名].zIndex
【プロパティ】
document.layers[親レイヤー名].layers[子レイヤー名].visibility
【プロパティ】






ネストされた子レイヤーの参照は、サンプルのように「document.layers[親レイヤー名].layers[子レイヤー名]」「document.layers.レイヤー名.layers.レイヤー名」または「document.layers[インデックス].layers[インデックス]」とすることによって可能です。
子レイヤーの場合は、「left」プロパティと「top」プロパティは親レイヤー左上角を基準とした、「pageX」プロパティと「pageY」プロパティはページ左上角を基準としたレイヤーの位置の値を、それぞれ持っています。
Layerオブジェクトは、Netscape Navigator 4.Xのみでサポートされています。

**Sample**

```
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<title></title>
</head>
<body bgcolor="#ffffff">
*子レイヤーの情報を取得する
<layer name="LAY1" left="200" top="50" width="200" height="150" z-index="1" bgcolor="#99ffcc">
    <layer name="LAY2" left="50" top="50" width="100" height="50" z-index="1" visibility="show" bgcolor="#ffffcc">
<b>子レイヤー</b>
    </layer>
</layer>
<p>
<script type="text/javascript">
<!--
document.write("オブジェクト名：",document.layers["LAY1"].layers["LAY2"].name);
document.write("<br>");
document.write("親レイヤー左からの位置：",document.layers.LAY1.layers.LAY2.left);
document.write("<br>");
document.write("親レイヤー上からの位置：",document.layers[0].layers[0].top);
document.write("<br>");
document.write("画面左からの位置：",document.layers["LAY1"].layers["LAY2"].pageX);
document.write("<br>");
document.write("画面上からの位置：",document.layers["LAY1"].layers["LAY2"].pageY);
document.write("<br>");
document.write("z-Indexの値：",document.layers["LAY1"].layers["LAY2"].zIndex);
document.write("<br>");
document.write("レイヤーの可視属性：",document.layers["LAY1"].layers["LAY2"].visibility);
//-->
</script>
</p>
</body>
</html>
```

JavaScript

# レイヤー上のどこでイベントが発生したかを取得する

イベント.layerX ←レイヤー上のX軸の値 [プロパティ]
イベント.layerY ←レイヤー上のY軸の値 [プロパティ]
オブジェクト.onmousedown = "スクリプト | 関数" [イベント]

N4.06
N4.X




eventオブジェクトの「layerX」プロパティと「layerY」プロパティは、イベントが起こったレイヤー上のX軸とY軸の値を持っています。
サンプルでは、レイヤー上でマウスボタンが押された時に、イベントタイプ「document.onmousedown」によってイベントが取得されて関数「eve1()」を発生し、イベントが持っているマウスカーソルのウィンドウとレイヤー上の位置の値を警告用のダイアログボックスに表示しています。
Layerオブジェクトは、Netscape Navigator 4.Xのみでサポートされています。

## Sample

```
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<title></title>
</head>
<body bgcolor="#ffffff">
*レイヤー上のどこでイベントが発生したかを取得する
<layer name="LAY1" visibility="show" bgcolor="blue" top=50 left=
100 width=200 height=150>

<script type="text/javascript">
<!--
function eve1(e) {
    alert ("イベント"+e.type +"が発生しました" +  "¥n" +
           "ウィンドウのX座標：" + e.pageX + "¥n" +
           "ウィンドウのY座標：" + e.pageY + "¥n" +
```

レイヤーを操作する

```
            "レイヤーのX座標：" + e.layerX + "¥n" +
            "レイヤーのY座標：" + e.layerY);
    return true;
}
document.onmousedown = eve1;
//-->
</script>

</layer>
</body>
</html>
```


イベントタイプ一覧：リファレンス「イベントタイプ」（P.622）

**重要**

## レイヤーの今後

レイヤーは、Netscape Navigator 4.0から追加された機能で、コンテンツの位置や重なりなどを細かく設定することができます。

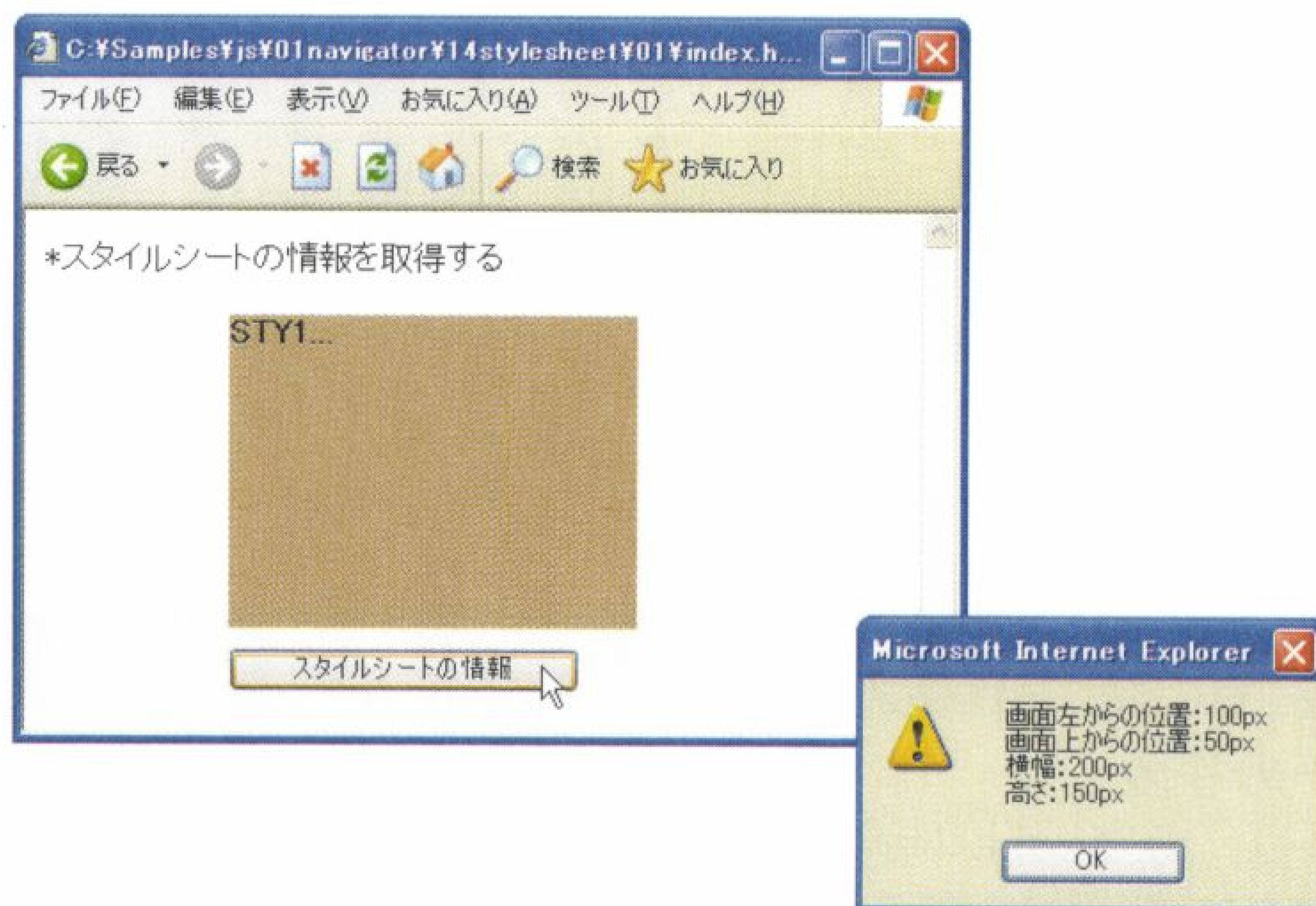
元々は、W3CでCSSの仕様がまだ決定されていない時期に、Netscape社がCSSの仕様として提案していた規格でしたが、その後CSSの仕様が確定した時には、レイヤーはそこには含まれていませんでした。このためレイヤーは、Netscape Navigatorの独自規格となってしまったのです。

これを受けて、標準仕様の完全準拠を目指して開発が進められていたMozillaでは、レイヤーのサポートを取りやめることになり、Mozillaを元に作られたNetscape 6.Xでも採用されていません。そして、おそらく今後も、レイヤーが再び採用されることはないと思われます。

確かにNetscape Navigator 4.XではCSSのサポートが中途半端で、コンテンツの位置や見映えを細かく設定したページを作るのは、なかなか大変です。しかし、今後レイヤーをサポートしたブラウザは出てこないであろうことや、Internet Explorerを始めとしたブラウザのCSS対応が一般的になって来ていることを考えると、今後は、レイヤーではなくCSSを使った方がよいでしょう。

# スタイルシートの情報を取得する

**document.all(オブジェクト名).style.left** [プロパティ]
**document.all(オブジェクト名).style.top** [プロパティ]
**document.all(オブジェクト名).style.width** [プロパティ]
**document.all(オブジェクト名).style.height** [プロパティ]
**document.getElementById(オブジェクト名).style.left** [プロパティ]
**document.getElementById(オブジェクト名).style.top** [プロパティ]
**document.getElementById(オブジェクト名).style.width** [プロパティ]
**document.getElementById(オブジェクト名).style.height** [プロパティ]
**document.layers[オブジェクト名].left** [プロパティ]
**document.layers[オブジェクト名].top** [プロパティ]




スタイルシートの各種の値は、Internet Explorerでは「document.all(オブジェクト名).style.プロパティ」で、Netscape Navigator 4.Xではレイヤーと同じ「document.layers[オブジェクト名].プロパティ」で、Netscape 6.0以降では「document.getElementById(オブジェクト名).style.プロパティ」の用法で取得可能です。
サンプルでは、まずInternet ExplorerかNetscape Navigatorかの判断を行っています。そして、ブラウザがInternet Explorerだった場合は、「document.all(オブジェクト名)」を使ってスタイルシートの値を取得しています。また、ブラウザがNetscape Navigatorだった場合は、さらにNetscape 6.0などのDOMに対応したブラウザか、レイヤーに対応したNetscape Navigator 4.Xのブラウザかの判断を行います。そして、それぞれに合わせて、DOMに対応したブラウザは「document.getElementById(オブジェクト名)」を、レイヤーに対応したブラウザは「document.layers[オブジェクト名]」を使ってスタイルシートの値を取得しています。

「left」プロパティはウィンドウの表示領域あるいは親スタイルシート左上角からのスタイルシート左上角の位置の値を、「top」プロパティはウィンドウの表示領域あるいは親スタイルシート左上角からのスタイルシート左上角の位置の値を、「width」プロパティはスタイルシートの横幅の値を、「height」プロパティはスタイルシートの高さの値を、それぞれ持っています。なお、「document.layers[オブジェクト名]」で得られる値は数値、「document.all(オブジェクト名)」や「document.getElementById(オブジェクト名)」で得られる値は文字列となります。

サンプルでは紹介していませんが、Internet Explorer 5.0 以降では、「document.getElementById(オブジェクト名)」の用法も使用可能です。また、サンプルのようにテキストのみのスタイルシートでは、Netscape Navigator 4.X ではスタイルシートの幅や高さの値を取得することができません。

なお、<meta> を入れてしまうと、Netscape Navigator 4.X ではテキストのみのスタイルシートが表示されない場合があるので、サンプルでは <meta> を省略しています。

## Sample

```
<html>
<head>
<title></title>

<script type="text/javascript">
<!--
function StyValue(){
    if(navigator.appName.charAt(0)=="M"){
        if(document.all){
            alert("画面左からの位置："+document.all("STY1").style.left+
"¥n"+
                "画面上からの位置："+document.all("STY1").style.top+
"¥n"+
                "横幅："+document.all("STY1").style.width+"¥n"+
                "高さ："+document.all("STY1").style.height);
        }
    }
    if(navigator.appName.charAt(0)=="N"){
        if(document.getElementById){
            alert("画面左からの位置："+document.getElementById("STY1")
.style.left+"¥n"+
                "画面上からの位置："+document.getElementById("STY1")
.style.top+"¥n"+
                "横幅："+document.getElementById("STY1").style.wi
dth+"¥n"+
                "高さ："+document.getElementById("STY1").style.he
ight);
        }
        if(document.layers){
            alert("画面左からの位置："+document.layers["STY1"].left+"¥n"+
                "画面上からの位置："+document.layers["STY1"].top);
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

```
            }
        }
}
//-->
</script>

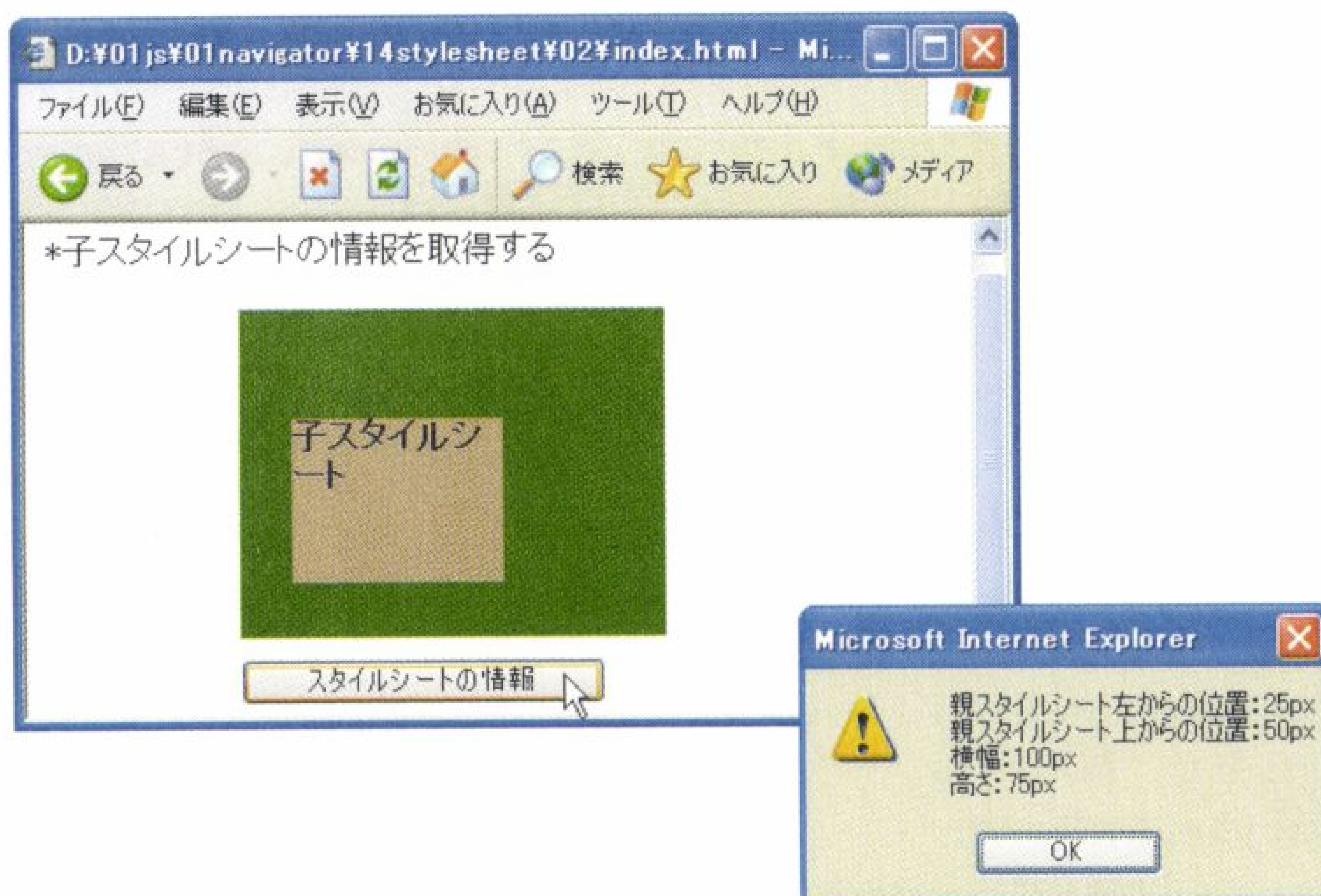
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
＊スタイルシートの情報を取得する
<div id="STY1" style="position:absolute; left:100px; top:50px; width:200px; height:150px;background:Tan">
<b>STY1...</b>
</div>
<div id="STY2" style="position:absolute; left:100px; top:210px">
<form>
    <input type="button" value=" スタイルシートの情報 " onClick="StyValue()">
</form>
</div>
</body>
</html>
```

<div>：「基本的な内容」の「目的に応じて範囲を設定する」(P.16)
navigator.appName：「navigatorオブジェクト」の「ブラウザ名を取得する」(P.310)
charAt()：「stringオブジェクト」の「n番目の文字を抜き出す」(P.555)
付録コラム「DynamicHTMLとは」(P.652)

# 子スタイルシートの情報を取得する

document.getElementById(オブジェクト名).style.left [プロパティ]
document.getElementById(オブジェクト名).style.top [プロパティ]
document.getElementById(オブジェクト名).style.width [プロパティ]
document.getElementById(オブジェクト名).style.height [プロパティ]
document.all(オブジェクト名).all(オブジェクト名).style.left [プロパティ]
document.all(オブジェクト名).all(オブジェクト名).style.top [プロパティ]
document.all(オブジェクト名).all(オブジェクト名).style.width [プロパティ]
document.all(オブジェクト名).all(オブジェクト名).style.height [プロパティ]




ネストされた子スタイルシートの参照は、「getElementById()」メソッドでは「document.getElementById(オブジェクト名).style.プロパティ」として直接オブジェクトを指定することができますが、「all()」メソッドの場合は「document.all(親スタイルシート名).all(子スタイルシート名).style.プロパティ」としてスタイルシートの階層に従って設定する必要があります。
サンプルでは、「getElementById()」メソッドに対応したInternet Explorer 5.0やNetscape 6.0以降のブラウザは「getElementById()」メソッドの用法を使って、「all()」メソッドに対応したInternet Explorer 4.Xのブラウザは「all()」メソッドの用法を使って、子スタイルシートの各種の値を取得しています。

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

Internet Explorer 4.Xでは、子スタイルシートの「width」プロパティの値は取得できません。また、Netscape Navigator 4.Xでは、ネストされたスタイルシートの扱い自体に問題があるため、サンプルではNetscape Navigator 4.Xの場合は何も処理をしないようにしています。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function StyValue(){
    if (navigator.appVersion.indexOf("MSIE 5") != -1 || navigator.
appVersion.indexOf("MSIE 6") != -1){
        if(document.getElementById){
            alert("親スタイルシート左からの位置："+document.getElementById
("STY2").style.left+"¥n"+
                        "親スタイルシート上からの位置："+document.getElementById
("STY2").style.top+"¥n"+
                        "横幅："+document.getElementById("STY2").style.wid
th+"¥n"+
                        "高さ："+document.getElementById("STY2").style.hei
ght);
        }
    }
    else{
        if(document.all){
            alert("親スタイルシート左からの位置："+document.all("STY1").
all("STY2").style.left+"¥n"+
                        "親スタイルシート上からの位置："+document.all("STY1").
all("STY2").style.top+"¥n"+
                        "横幅："+document.all("STY1").all("STY2").width
+"¥n"+
                        "高さ："+document.all("STY1").all("STY2").style.
height);
        }
    }
    if(navigator.appName.charAt(0)=="N"){
        if(document.getElementById){
            alert("親スタイルシート左からの位置："+document.getElementBy
Id("STY2").style.left+"¥n"+
                        "親スタイルシート上からの位置："+document.getElementBy
Id("STY2").style.top+"¥n"+
```

```
                    "横幅："+document.getElementById("STY2").style.
width+"¥n"+
                    "高さ："+document.getElementById("STY2").style.
height);
        }
    }
}
//-->
</script>

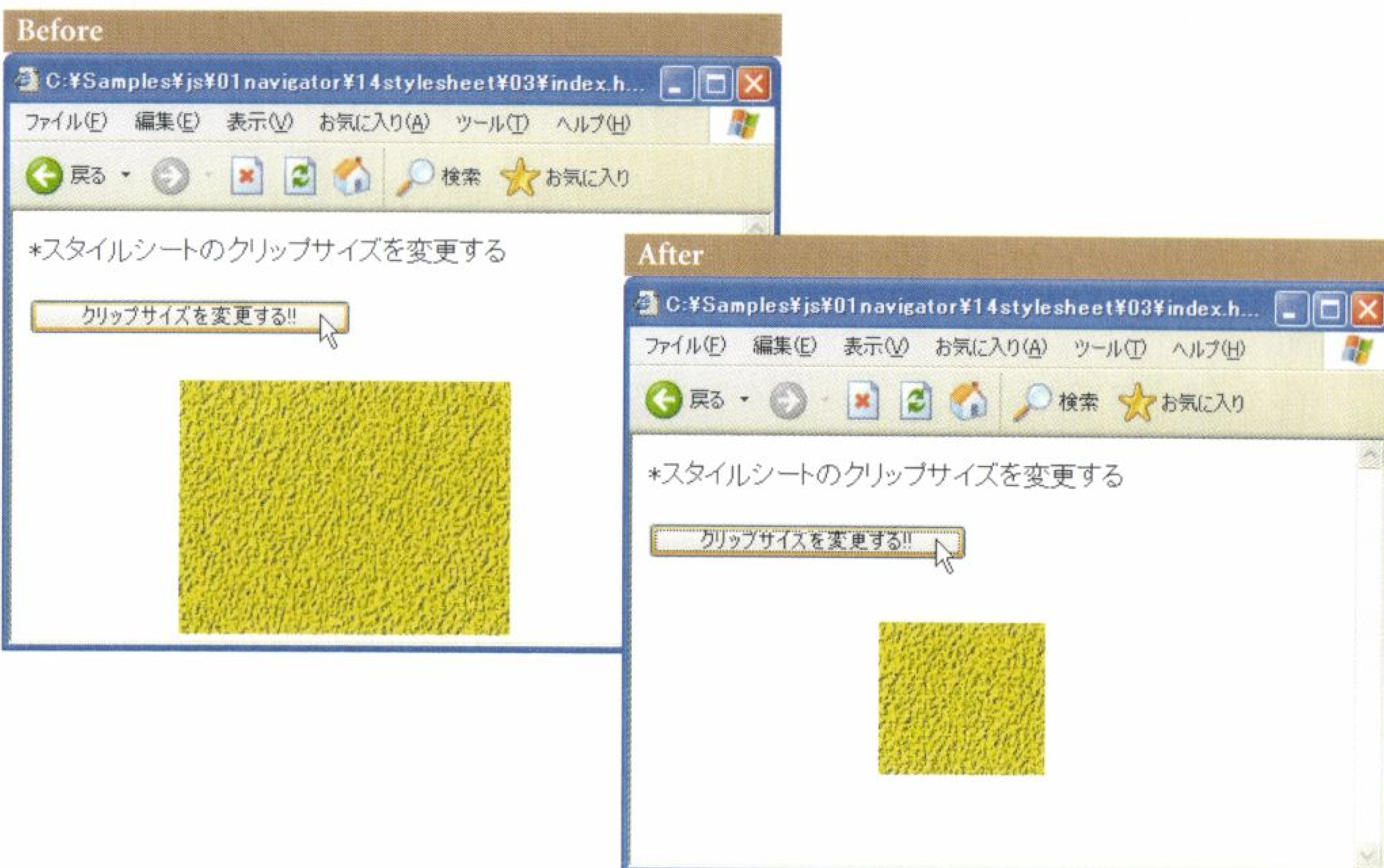
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*子スタイルシートの情報を取得する
<div id="STY1" style="position:absolute; left:100px; top:50px; wid
th:200px; height:150px; z-index=1;visibility:visible; background:
Green">
    <div id="STY2" style="position:absolute; left:25px; top:50px;
width:100px; height:75px; z-index=2;visibility:visible; background
: Tan">
<b>子スタイルシート</b>
    </div>
</div>
<div id="STY3" style="position:absolute; left:100px; top:210px">
<form>
    <input type="button" value=" スタイルシートの情報 " onClick="StyVal
ue()">
</form>
</div>
</body>
</html>
```


<div>：「基本的な内容」の「目的に応じて範囲を設定する」(P.16)
navigator.appVersion：「navigatorオブジェクト」の「ブラウザのバージョンを取得する」(P.311)
indexOf()：「stringオブジェクト」の「先頭から文字列を検索する」(P.558)
付録コラム「DynamicHTMLとは」(P.652)

# スタイルシートのクリップサイズを変更する

document.getElementById(オブジェクト名).style.clip="rect(上,右,下,左)" [プロパティ]
document.all(オブジェクト名).style.clip="rect(上,右,下,左)" [プロパティ]
document.layers[オブジェクト名].clip.top= ピクセル [プロパティ]
document.layers[オブジェクト名].clip.right= ピクセル [プロパティ]
document.layers[オブジェクト名].clip.bottom= ピクセル [プロパティ]
document.layers[オブジェクト名].clip.left= ピクセル [プロパティ]




サンプルでは、[クリップサイズを変更する!!]ボタンが押された時に、関数「Size()」が発生し、関数内の処理でスタイルシートのクリップサイズを変更することにより、スタイルシートの表示領域を変更しています。
関数「Size()」内では、if文を使ってブラウザの種類やバージョンなどを判断します。Internet Explorer 5.0以降やNetscape 6.0などの「getElementById()」メソッドに対応したブラウザは「document.getElementById(オブジェクト名).style.clip」を、Internet Explorer 4.0などの「all()」メソッドに対応したブラウザは「document.all(オブジェクト名).style.clip」を、Netscape Navigator 4.Xのレイヤーに対応したブラウザは「document.layers[オブジェクト名]」を、使用するようにしています。
「getElementById()」メソッドや「all()」メソッドは、「clip」プロパティを使用して、その値で「"rect(クリップ上の位置,クリップ右の位置,クリップ下の位置,クリップ左の位置)"」として設定します。また、スタイルシートに対応したNetscape Navigator 4.Xでは、ク

リップ上の位置は「clip.top」プロパティ、クリップ右の位置は「clip.right」プロパティ、クリップ下の位置は「clip.bottom」プロパテイ、クリップ左の位置は「clip.left」プロパティで設定しています。
なお、Netscape Navigator 4.X ではスタイルシートの表示に問題があるので、スタイルシートと同じ大きさの画像を設定し、スタイルシート全体が表示されるようにしています。
また、このサンプルは、Macintosh版のInternet Explorer 4.0では機能しません。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function Size(){
   if(navigator.appName.charAt(0)=="M"){
        if (navigator.appVersion.indexOf("MSIE 5") != -1 || 
navigator.appVersion.indexOf("MSIE 6") != -1){
            if(document.getElementById){
                document.getElementById("STY1").style.clip="rect(
10,150,100,50)"
            }
        }
        else{
            if(document.all){
                document.all("STY1").style.clip="rect(10,150,100,
50)"
            }
        }
    }
    if(navigator.appName.charAt(0)=="N"){
        if(document.getElementById){
            document.getElementById("STY1").style.clip="rect(10,
150,100,50)"
        }
        if(document.layers){
            document.layers["STY1"].clip.top=10;
            document.layers["STY1"].clip.right=150;
            document.layers["STY1"].clip.bottom=100;
            document.layers["STY1"].clip.left=50;
        }
    }
}
```




```
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
＊スタイルシートのクリップサイズを変更する
<form>
    <input type="button" value="クリップサイズを変更する!!" onclick="Siz
e()">
</form>
<div id="STY1" style="position:absolute; width:200px; height:150
px; left: 100px; top:100px">
<img src="image.jpg" alt="image.gif" width="200" height="150">
</div>
</body>
</html>
```

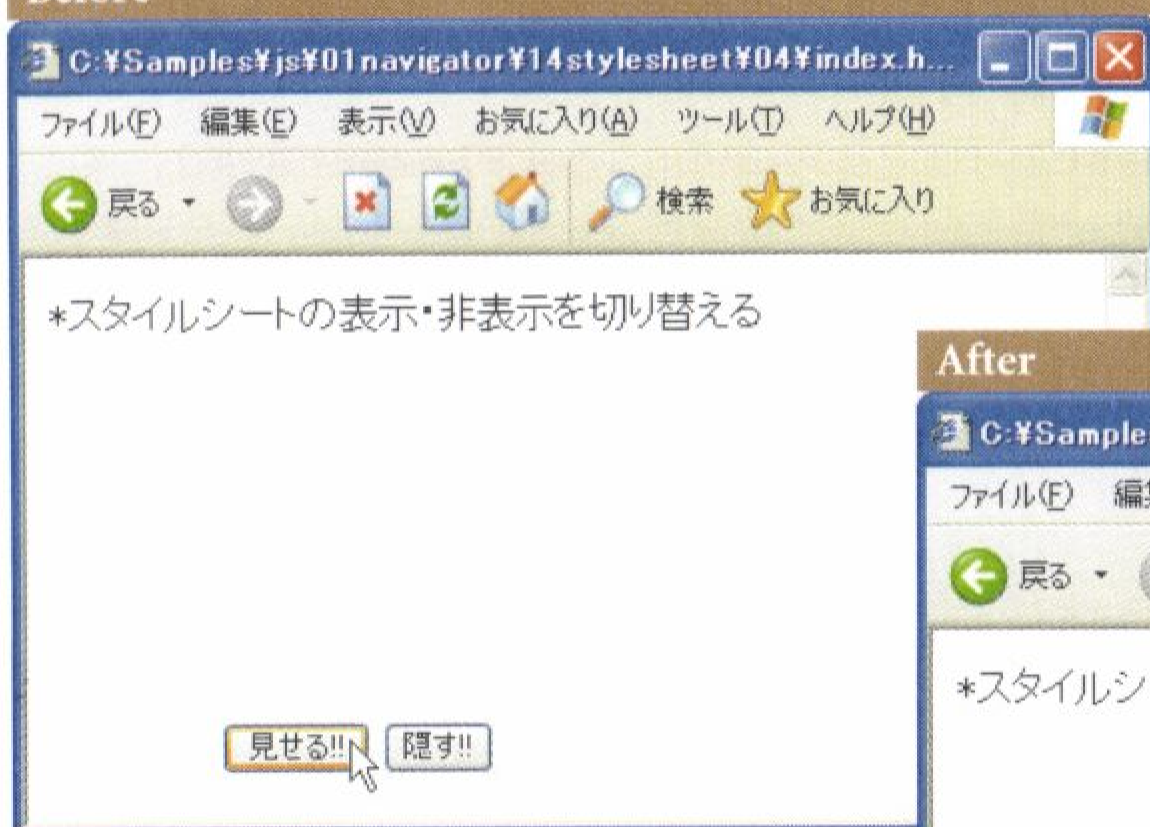
<div>：「基本的な内容」の「目的に応じて範囲を設定する」(P.16)
navigator.appName：「navigatorオブジェクト」の「ブラウザ名を取得する」(P.310)
navigator.appVersion：「navigatorオブジェクト」の「ブラウザのバージョンを取得する」(P.311)
charAt()：「stringオブジェクト」の「n番目の文字を抜き出す」(P.555)
indexOf()：「stringオブジェクト」の「先頭から文字列を検索する」(P.558)
付録コラム「DynamicHTMLとは」(P.652)

# スタイルシートの表示・非表示を切り替える

**document.getElementById(*オブジェクト名*).style.visibility="*visible | hidden*"** ［プロパティ］
**document.all(*オブジェクト名*).style.visibility="*visible | hidden*"** ［プロパティ］
**document.layers[*オブジェクト名*].visibility="*visible,show | hidden,hide*"** ［プロパティ］



Before




After




スタイルシートの可視属性を設定する「visibility」の値は、Netscape Navigator 4.Xのスタイルシートに対応したブラウザで使用する「layers」の場合、表示状態は「show」、非表示状態は「hidden」または「hide」です。Internet ExplorerやNetscape 6.0で使用する「getElementById()」メソッドや「all()」メソッドの場合、表示状態は「visible」、非表示状態は「hidden」となります。
サンプルでは、はじめにスタイルシートの「visibility」の設定を「hidden」として、スタイルシートを非表示状態にしています。そして、[見せる!!]ボタンが押された時に発生する関数「SImg()」内の処理によって、「document.getElementById("STY1").style.visibility」や「document.all("STY1").style.visibility」には「visible」の値を設定し、「document.layers["STY1"].visibility」には「show」の値を設定することによって、表示状態にします。[隠す!!]ボタンが押された時は、関数「HImg()」内の処理によって、「document.getElementById("STY1").style.visibility」や「document.all("STY1").style.visibility」には「hidden」の値を、「document.layers["STY1"].visibility」には、「hide」の値を設定することによって非表示状態にしています。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function SImg(){
    if(navigator.appName.charAt(0)=="M"){
        if (navigator.appVersion.indexOf("MSIE 5") != -1 || naviga
tor.appVersion.indexOf("MSIE 6") != -1){
            if(document.getElementById){document.getElementById(
"STY1").style.visibility="visible"}
        }
        else{
            if(document.all){document.all("STY1").style.visibility
="visible"}
        }
    }
    if(navigator.appName.charAt(0)=="N"){
        if(document.getElementById){document.getElementById("STY1"
).style.visibility="visible"}
        if(document.layers){document.layers["STY1"].visibility=
"show"}
    }
}
function HImg(){
    if(navigator.appName.charAt(0)=="M"){
        if (navigator.appVersion.indexOf("MSIE 5") != -1 || naviga
tor.appVersion.indexOf("MSIE 6") != -1){
            if(document.getElementById){document.getElementById(
"STY1").style.visibility="hidden"}
        }
        else{
            if(document.all){document.all("STY1").style.visibility
="hidden"}
        }
    }
    if(navigator.appName.charAt(0)=="N"){
        if(document.getElementById){document.getElementById("STY1"
).style.visibility="hidden"}
        if(document.layers){document.layers["STY1"].visibility=
"hide"}
    }
}
```

```
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
＊スタイルシートの表示・非表示を切り替える
<div id="STY1" style="position:absolute; visibility:hidden; left:100px; top:50px">
<img src="image.jpg" alt="image.jpg" width="100" height="100">
</div>
<div id="STY2" style=" position: absolute; top: 185px; left: 80px; visibility: visible">
<form>
    <input type="button" value="見せる!!" onclick="SImg()">
    <input type="button" value="隠す!!" onclick="HImg()">
</form>
</div>
</body>
</html>
```


<div>：「基本的な内容」の「目的に応じて範囲を設定する」(P.16)
navigator.appName：「navigatorオブジェクト」の「ブラウザ名を取得する」(P.310)
navigator.appVersion：「navigatorオブジェクト」の「ブラウザのバージョンを取得する」(P.311)
charAt()：「stringオブジェクト」の「n番目の文字を抜き出す」(P.555)
indexOf()：「stringオブジェクト」の「先頭から文字列を検索する」(P.558)
付録コラム「DynamicHTMLとは」(P.652)

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# スタイルシートを移動する

**document.getElementById(オブジェクト名).style.left="ピクセル"** [プロパティ]
**document.all(オブジェクト名).style.left="ピクセル"** [プロパティ]
**document.layers[オブジェクト名].left="ピクセル"** [プロパティ]










スタイルシート左側のウィンドウ表示領域の左端からの位置は、Internet Explorer 5.0以降やNetscape 6.0などの「getElementById()」メソッドに対応したブラウザでは「document.getElementById(オブジェクト名).style.left」で、Internet Explorer 4.0などの「all()」メソッドに対応したブラウザでは「document.all(オブジェクト名).style.left 」で、Netscape Navigator 4.Xのレイヤーに対応したブラウザは「document.layers[オブジェクト名].left 」で、設定することができます。

サンプルでは、まずウィンドウ左端から200ピクセルの位置にスタイルシートを設定しています。そして、ページが読み込まれた時にイベントハンドラ「onload」が関数「StyMove(){」を発生させ、関数内の処理で定期的にスタイルシートの左端の位置の値を変化させることによって、スタイルシートを左右に移動させています。

Netscape Navigator 4.Xのようなレイヤーに対応したブラウザでは、「document.layers[オブジェクト名].left」の値は数値です。ですから、「document.layers["STY1"].left -= 10」としてスタイルシート「STY1」を左に10ピクセル、「document.layers["STY3"].left += 10」として「STY3」を右に10ピクセルずらします。そして次の「setTimeout("StyMove()",100)」の処理で、再び関数「StyMove()」を発生さることによって、条件式「

スタイルシートを利用する

document.layers["STY1"].left >= 50」と「document.layers["STY3"].left <= 240」が真になるまで、この処理を続けるようにしています。
また、「document.getElementById(オブジェクト名).style.left」や「document.all(オブジェクト名).style.left」の値は、「"n px"」の形式の文字列です。サンプルでは、初期値を「200」とした変数「STY1_Point」と「STY3_Point」を、「STY1_Point-=10」や「STY3_Point+=10」として値を変化させ、それに文字列「px」を追加したものを、それぞれのスタイルシート左端の位置の値に設定しています。そして、スタイルシートの位置を設定した後は、レイヤーと同じように「setTimeout("StyMove()",100)」によって再び関数を発生させ、スタイルシート「STY1」がウィンドウ左端から50ピクセル、スタイルシート「STY3」がウィンドウ左端から240ピクセルになるまで、この処理を続けるようにしています。
なお、<meta>を入れると、Netscape Navigator 4.Xではテキストのみのスタイルシートが表示されない場合があるので、サンプルでは<meta>を省略しています。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title></title>
<script type="text/javascript">
<!--
var STY1_Point=200;
var STY3_Point=200;
function StyMove(){
    if(navigator.appName.charAt(0)=="M"){
        if (navigator.appVersion.indexOf("MSIE 5") != -1 || navigat
or.appVersion.indexOf("MSIE 6") != -1){
            if(document.getElementById){
                if (document.getElementById("STY1").style.left !=
"50px"){ document.getElementById("STY1").style.left = (STY1_Point-
=10)+"px" }
                if (document.getElementById("STY3").style.left !=
"240px"){ document.getElementById("STY3").style.left = (STY3_Point
+=10)+"px"}
                setTimeout("StyMove()",100);
            }
    }
        else{
            if(document.all){
                if (document.all("STY1").style.left != "50px"){
document.all("STY1").style.left = (STY1_Point-=10)+"px" }
                if (document.all("STY3").style.left != "240px"){
document.all("STY3").style.left = (STY3_Point+=10)+"px"}
                setTimeout("StyMove()",100);
            }
        }
    }
    if(navigator.appName.charAt(0)=="N"){
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

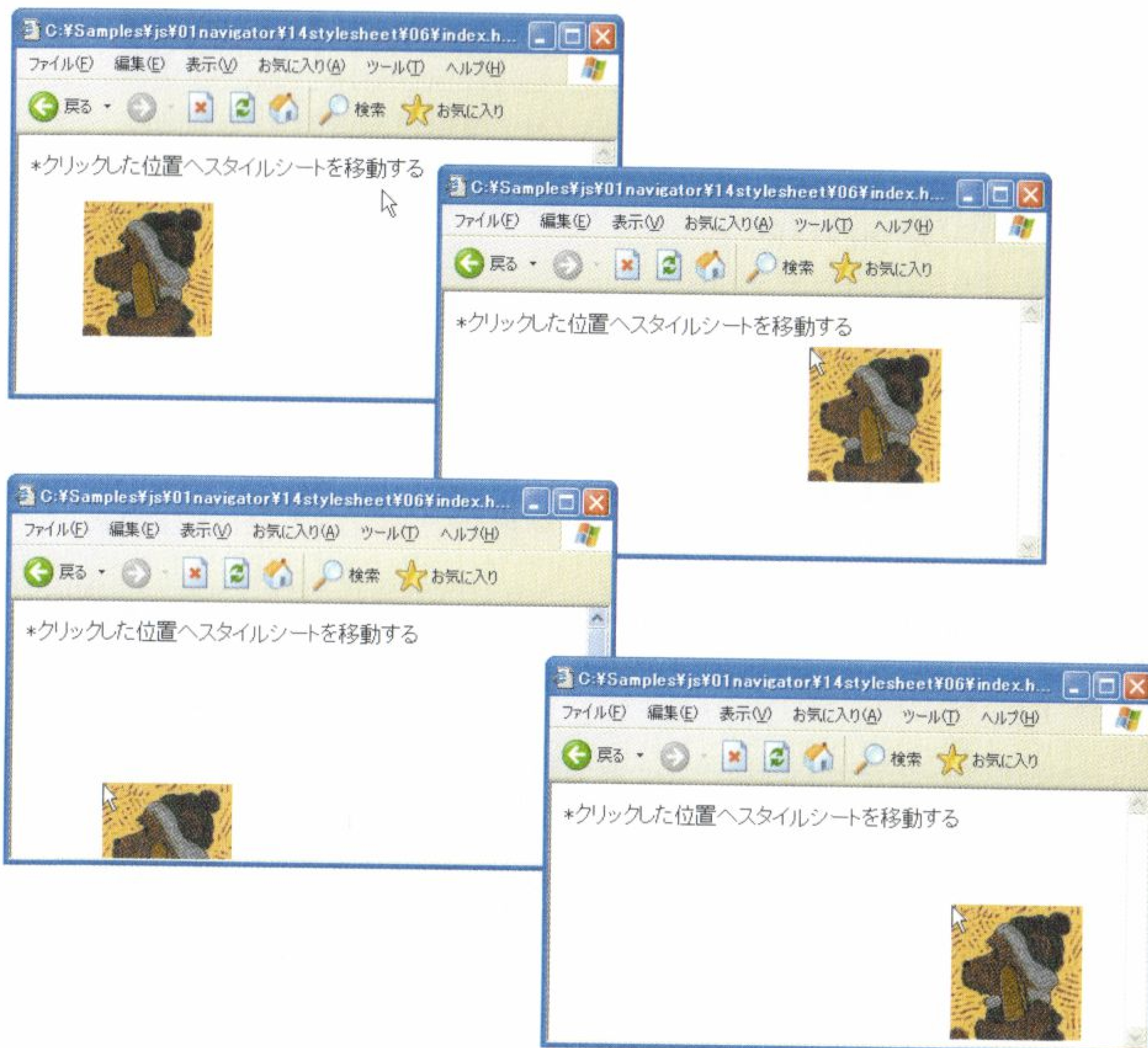
```
        if(document.getElementById){
            if (document.getElementById("STY1").style.left != "50p
x"){ document.getElementById("STY1").style.left = (STY1_Point-=10)
+"px" }
            if (document.getElementById("STY3").style.left != "240
px"){ document.getElementById("STY3").style.left = (STY3_Point+=
10)+"px"}
            setTimeout("StyMove()",100);
        }
        if(document.layers){
            if ( document.layers["STY1"].left >= 50 ){ document.la
yers["STY1"].left -= 10  }
            if( document.layers["STY3"].left <= 240 ){ document.la
yers["STY3"].left += 10 }
           setTimeout("StyMove()",100);
       }
    }
}
//-->
</script>
 ～中略～
</head>
<body onLoad="StyMove()">
*スタイルシートを移動する
<div id="STY1" style="position:absolute;left:200px; top:50px;font-
size:24px">
Welcome To
</div>
<div id="STY2" style="position:absolute;left:200px; top:50px;font-
size:24px">
MY
</div>
<div id="STY3" style="position:absolute;left:200px; top:50px;font-
size:24px">
Home Page
</div>
</body>
</html>
```

<div>：「基本的な内容」の「目的に応じて範囲を設定する」(P.16)
navigator.appName：「navigatorオブジェクト」の「ブラウザ名を取得する」(P.310)
navigator.appVersion：「navigatorオブジェクト」の「ブラウザのバージョンを取得する」(P.311)
charAt()：「stringオブジェクト」の「n番目の文字を抜き出す」(P.555)
indexOf()：「stringオブジェクト」の「先頭から文字列を検索する」(P.558)
付録コラム「DynamicHTMLとは」(P.652)

# クリックした位置へスタイルシートを移動する

**document.getElementById(オブジェクト名).style.left="ピクセル"** [プロパティ]
**document.getElementById(オブジェクト名).style.top="ピクセル"** [プロパティ]
**document.all(オブジェクト名).style.left="ピクセル"** [プロパティ]
**document.all(オブジェクト名).style.top="ピクセル"** [プロパティ]
**document.layers[オブジェクト名].left="ピクセル"** [プロパティ]
**document.layers[オブジェクト名].top="ピクセル"** [プロパティ]






サンプルでは、まず「document.onmousedown」を使って、ウィンドウ上のマウスボタンが押された時のイベントを取得して関数「eve()」を発生させます。関数の処理で、イベントが発生した位置をスタイルシートの位置の値に設定することによって、クリックした位置へスタイルシートを移動させています。「getElementById()」メソッドや「all()」メソッド、レイヤーのどの用法でも、ウィンドウ表示領域左端からのスタイルシート左端の位置は「left」プロパティ、ウインドウ表示領域上からのスタイルシート上の位置は「top」プロパティで設定することができます。

移動は、関数内の処理で、まずif文を使ってブラウザやバージョンを判断した後、それぞれに合わせて「document.getElementById(オブジェクト名).style」や「document.all(オブジェクト名).style」、「document.layers[オブジェクト名]」を使って「left」プロパティと「top」プロパティに、イベントが発生した位置の値を設定しています。
イベントが発生した位置の値を表す用法は、Internet Explorerでは上からの位置が「window.event.offsetX」、左からの位置が「window.event.offsetY」、Netscape Navigatorでは上からの位置が「引数.pageX」、左からの位置が「引数.pageY」となります。Internet ExplorerとNetscape Navigatorでは違う点に注意してください。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function eve(e) {
    if(navigator.appName.charAt(0)=="M"){
        if (navigator.appVersion.indexOf("MSIE 5") != -1 ||
navigator.appVersion.indexOf("MSIE 6") != -1){
            if(document.getElementById){
                document.getElementById("STY").style.left=window.
event.offsetX;
                document.getElementById("STY").style.top=window.
event.offsetY;
            }
        }
        else{
            if(document.all){
                document.all("STY").style.left=window.event.offsetX;
                document.all("STY").style.top=window.event.offsetY;
            }
        }
    }
    if(navigator.appName.charAt(0)=="N"){
        if(document.getElementById){
            document.getElementById("STY").style.left=e.pageX;
            document.getElementById("STY").style.top=e.pageY;
        }
        if(document.layers){
            document.layers["STY"].left=e.pageX;
            document.layers["STY"].top=e.pageY;
        }
    }
```

```
}
document.onmousedown = eve;
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*クリックした位置へスタイルシートを移動する
<div id="STY" style="position:absolute; width:100px; height:100px; ↵
left:50px; top:50px">
<img src="image.jpg" alt="image.jpg" width="100" height="100">
</div>
</body>
</html>
```

 <div>：「基本的な内容」の「目的に応じて範囲を設定する」(P.16)
navigator.appName：「navigatorオブジェクト」の「ブラウザ名を取得する」(P.310)
navigator.appVersion：「navigatorオブジェクト」の「ブラウザのバージョンを取得する」(P.311)
charAt()：「stringオブジェクト」の「n番目の文字を抜き出す」(P.555)
indexOf()：「stringオブジェクト」の「先頭から文字列を検索する」(P.558)
イベントタイプ一覧：リファレンス「5. イベントタイプ」(P.622)
付録コラム「DynamicHTMLとは」(P.652)

JavaScript

# 年・月・日・時・分・秒を表示する

オブジェクト名 = new Date()
オブジェクト名.getYear() [メソッド]
オブジェクト名.getDate() [メソッド]
オブジェクト名.getMonth() [メソッド]
オブジェクト名.getHours() [メソッド]
オブジェクト名.getMinutes() [メソッド]
オブジェクト名.getSeconds() [メソッド]




サンプルでは、「now = new Date()」でマシンのシステム時計から現在時刻の要素を取り出したオブジェクト「now」を作成し、そこからメソッドを使って各時間の要素を取得しています。

「getYear()」メソッドは西暦の下2桁の数値を、「getMonth()」メソッドは1月を0とした月に応じた0から11までの数値を、「getDate()」メソッドは日に応じた1から31までの数値を、「getHours()」メソッドは時間に対応した0から23までの数値を、「getMinutes()」メソッドは分に対応した0から59までの数値を、「getSeconds()」メソッドは秒に対応した0から59までの数値を、それぞれ取得します。

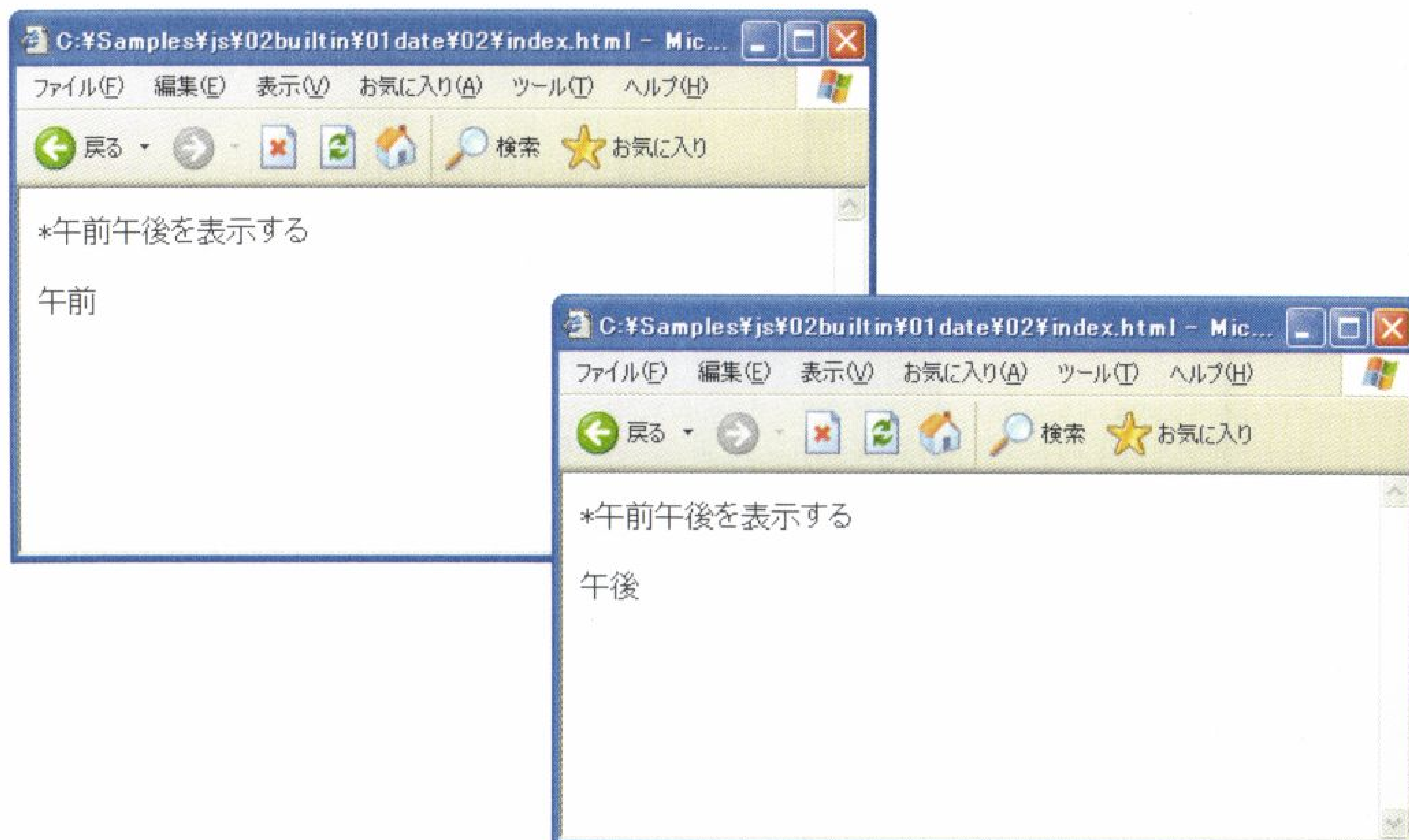
サンプルでは、2000年以降でも、「getYear()」メソッドを使って正確な4桁の年号を表示できるようにしています。詳しい解説は、付録コラムの「Javascriptの2000年問題」(P.650)を参照してください。

## Sample

```
<script type="text/javascript">
<!--
now = new Date();
    if ( now.getYear() >= 2000 ){ document.write(now.getYear(),"年
") }
    else { document.write(now.getYear()+1900,"年") }
    document.write(now.getMonth()+1,"月",now.getDate(),"日");
    document.write(now.getHours(),"時",now.getMinutes(),"分");
    document.write(now.getSeconds(),"秒");
//-->
</script>
```

# 午前午後を表示する

**オブジェクト名 = new Date()**
**オブジェクト名.getHours()** [メソッド]




サンプルでは、「getHours()」メソッドで現在時間を取得し、その数値が12より小さければ「午前」、大きければ「午後」と表示します。
「条件式 ? x:y」は、条件式が真の場合は「x」の値を、偽の場合は「y」の値を返します。

## Sample

```
<script type="text/javascript">
<!--
var now = new Date();
var AMPM = now.getHours();
    document.write( AMPM<12 ? "午前":"午後");
//-->
</script>
```

### 注意

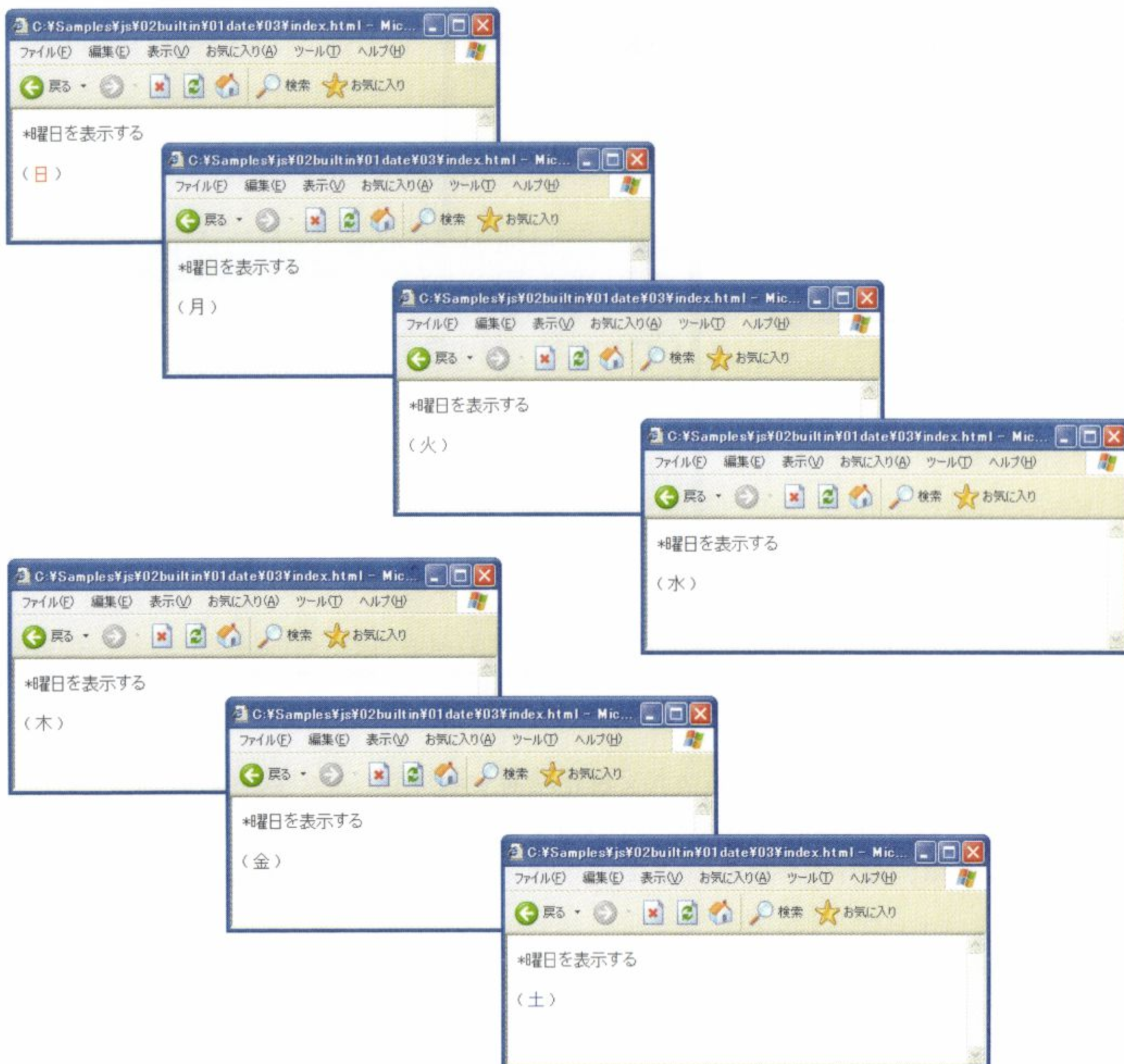
**月の値を表示する時の注意**

月の値を返す「getMonth()」メソッドは、実際の月より1小さい数値を返します。正確に月を表示させたい時には、「getMonth()+1」と1を加えることを忘れないようにしてください。

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 曜日を表示する

**オブジェクト名 = new Date()**
**オブジェクト名.getDay()** [メソッド]




曜日を取得する「getDay()」メソッドは、日曜日の場合は0、月曜日の場合は1、という順番で、曜日の値を0から6までの数値で取得します。
サンプルでは、取得した数値をif文で参照し、日曜日は赤い文字で、土曜日は青い文字で、その他の曜日はフォントの色指定なしで書き出しています。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
```

```
<script type="text/javascript">
<!--
var y0 = "日";
var y1 = "月";
var y2 = "火";
var y3 = "水";
var y4 = "木";
var y5 = "金";
var y6 = "土";
function gety(y){
    if (y==0) { document.write( y0.fontcolor("red") ) }
    if (y==1) { document.write( y1 ) }
    if (y==2) { document.write( y2 ) }
    if (y==3) { document.write( y3 ) }
    if (y==4) { document.write( y4 ) }
    if (y==5) { document.write( y5 ) }
    if (y==6) { document.write( y6.fontcolor("blue") ) }
}
//-->
</script>
 ～中略～
</head>
<body>
*曜日を表示する
<p>
(
<script type="text/javascript">
<!--
    day = new Date();
    gety(day.getDay());
//-->
</script>
)
</p>
</body></html>
```

string.fontcolor()：「stringオブジェクト」の「文字色を指定する」(P.545)

注意

### Macintosh版のNetscape Navigatorで Dateオブジェクトを使う時の注意

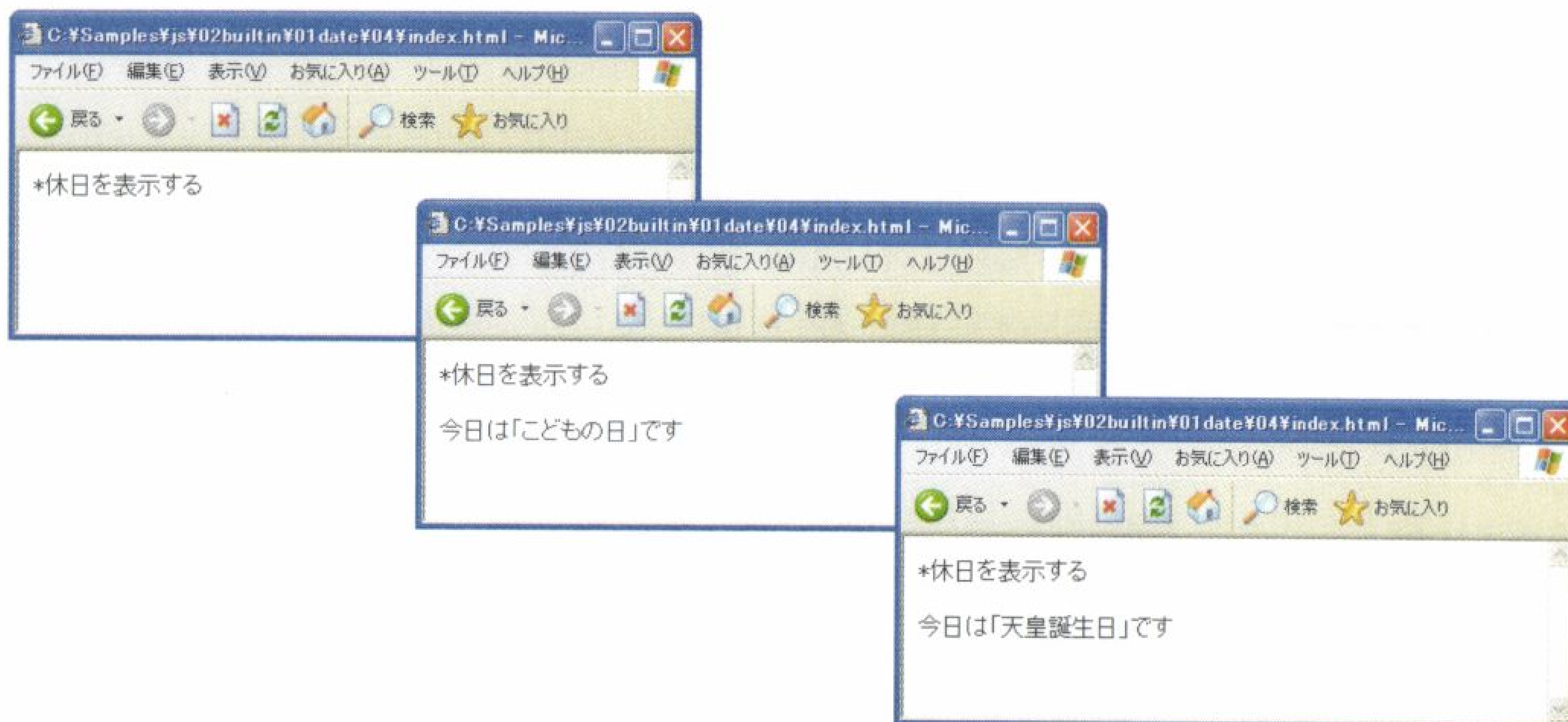
Macintosh版のNetscape Navigator 4.X以前では、グリニッジ標準時(GTM)を扱う「toGMTString()」メソッドや「getTimezoneOffset()」メソッドは、正しい時間を返しません。

また、「getTime()」「parse()」「UTC()」などの、1970年1月1日0時0分0秒からの経過時間を取り扱うメソッドも、正しい日付を返さない場合があります。

# 休日を表示する

オブジェクト名 = new Date()
オブジェクト名.getMonth() [メソッド]
オブジェクト名.getDate() [メソッド]
オブジェクト名.getDay [メソッド]






サンプルでは、HTMLファイルが読み込まれた時に、関数「gethd()」内の「holiday.getMonth()+1」で月の値を、「holiday.getDate()」で日の値を、「holiday.getDay()」で曜日の値を、それぞれ取得します。それらの値を「function gethd(m,d,y){～}」内のif文で参照し、休日であれば休日名を書き出します。
「function gethd(m,d,y)」のmには月の値が、dには日の値が、yには曜日の値が、それぞれ渡されます。そして、if文内の条件式で、たとえば「(m==1&&d==1)」は「月の値が1でかつ日の値が1である」ことを表します。この条件式が真の時、つまり1月1日の時には、「var」によって「hd0」と宣言された変数の値、「今日は「元日」です!!明けましておめでとうございます」という文字列が書き出されます。
また、成人の日は1月の第2月曜日、体育の日は10月の第2月曜日、つまりそれぞれ1月8日から14日の間の月曜日、10月8日から14日の間の月曜日です。そのためif文内の条件式を、成人の日は「(m==1&&8<=d&&d<=14&&y==1)」、体育の日は「(m==10&&8<=d&&d<=14&&y==1)」として、休日を特定しています。同様に、海の日は7月の、敬老の日は9月の第3月曜日です。つまり、それぞれの月の15日から21日の間の月曜日を特定することによって、休日を判断しています。



## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
```

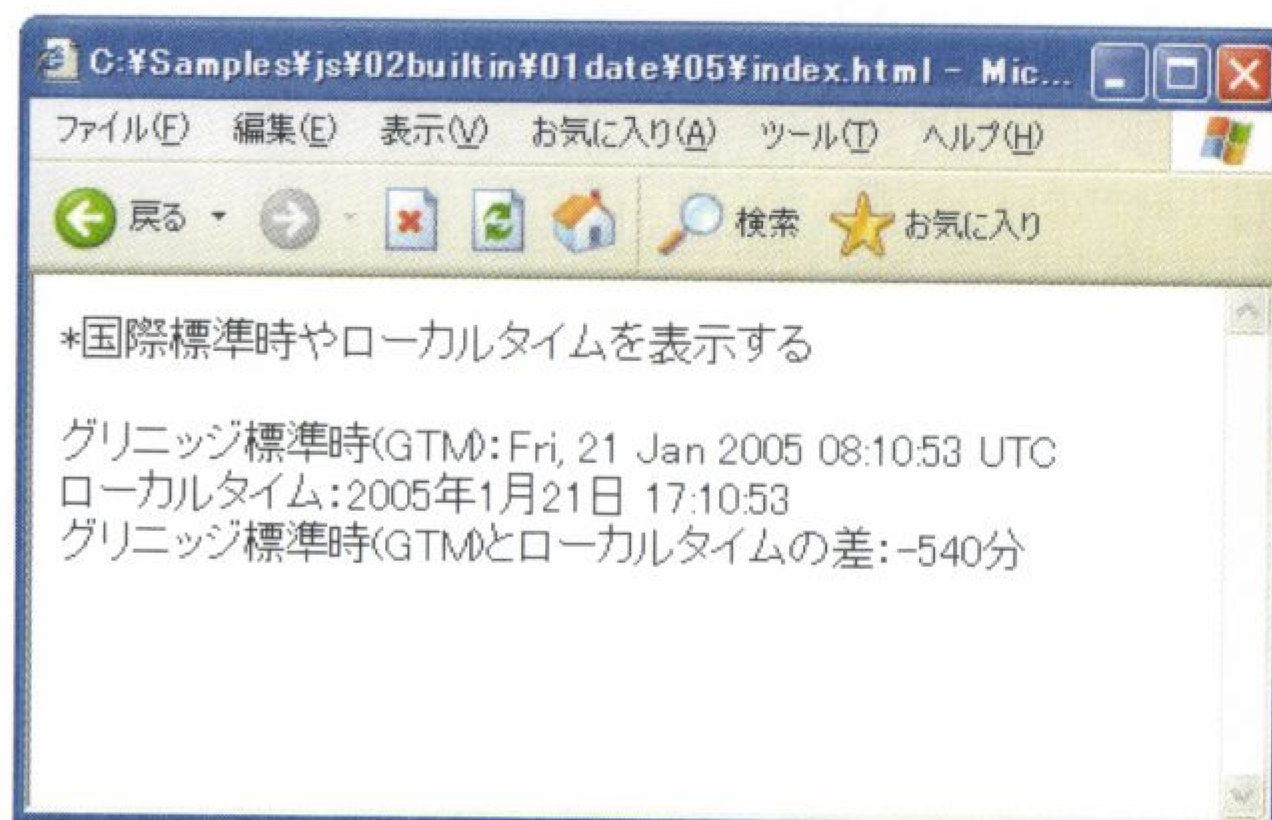
```
<title></title>
<script type="text/javascript">
<!--
var hd0 = "今日は「元日」です!!明けましておめでとうございます";
var hd1 = "今日は「成人の日」です";
var hd2 = "今日は「建国記念日」です";
var hd3 = "今日は「春分の日」です";
var hd4 = "今日は「みどりの日」です";
var hd5 = "今日は「憲法記念日」です";
var hd6 = "今日は「国民の休日」です";
var hd7 = "今日は「こどもの日」です";
var hd8 = "今日は「海の日」です";
var hd9 = "今日は「敬老の日」です";
var hd10 = "今日は「秋分の日」です";
var hd11 = "今日は「体育の日」です";
var hd12 = "今日は「文化の日」です";
var hd13 = "今日は「勤労感謝の日」です";
var hd14 = "今日は「天皇誕生日」です";
function gethd(m,d,y){
if (m==1&&d==1) { document.write( hd0 ) }
if (m==1&&8<=d&&d<=14&&y==1) { document.write( hd1 ) }
if (m==2&&d==11) { document.write( hd2 ) }
if (m==3&&d==20) { document.write( hd3 ) }
if (m==4&&d==29) { document.write( hd4 ) }
if (m==5&&d==3) { document.write( hd5 ) }
if (m==5&&d==4) { document.write( hd6 ) }
if (m==5&&d==5) { document.write( hd7 ) }
if (m==7&&15<=d&&d<=21&&y==1) { document.write( hd8 ) }
if (m==9&&15<=d&&d<=21&&y==1) { document.write( hd9 ) }
if (m==9&&d==23) { document.write( hd10 ) }
if (m==10&&8<=d&&d<=14&&y==1) { document.write( hd11 ) }
if (m==11&&d==3) { document.write( hd12 ) }
if (m==11&&d==23) { document.write( hd13 ) }
if (m==12&&d==23) { document.write( hd14) }
}
//-->
</script>
 ～中略～
<body>
*休日を表示する
<p>
<script type="text/javascript">
<!--
holiday = new Date();
gethd(holiday.getMonth()+1,holiday.getDate(),holiday.getDay());
//-->
</script>
</p>
</body></html>
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 国際標準時やローカルタイムを表示する

オブジェクト名 = new Date()
オブジェクト名.toGMTString() [メソッド]
オブジェクト名.toLocaleString() [メソッド]
オブジェクト名.getTimezoneOffset() [メソッド]




「toGMTString()」メソッドは日付と時間をGTM形式の文字列に変換し、「toLocaleString()」メソッドは日付と時間をローカルタイムの文字列に変換します。

「toGMTString()」メソッドは実行されたマシンの環境によって、「toLocaleString()」メソッドは実行された地域とマシンの環境よって結果が違います。

「getTimezoneOffset()」メソッドは、グリニッジ標準時とローカルタイムの差を分で返します。

## Sample

```
<script type="text/javascript">
<!--
    gtm = new Date();
    document.write("グリニッジ標準時(GTM):",gtm.toGMTString());
    document.write("<br>");
    document.write("ローカルタイム:",gtm.toLocaleString());
    document.write("<br>");
    document.write("グリニッジ標準時(GTM)とローカルタイムの差:",gtm.get
TimezoneOffset(),"分");
//-->
</script>
```

# 日時を後から変更する

オブジェクト名 = new Date("月 日, 年 時:分:秒")
オブジェクト名.getTime() [メソッド]
オブジェクト名.setTime() [メソッド]






「setTime()」メソッドは、ミリ秒単位で日付と時間の設定を行います。
サンプルでは、「ChangeDay = new Date("may 2, 1967 23:45:00")」で1967年5月2日23時45分の要素を持った「ChangeDay」というオブジェクトを作ります。その後、「setTime()」メソッドを使用して、「ChangeDay」オブジェクトを31日後の日時を持つオブジェクトにセットし直しています。



## Sample

```
<script type="text/javascript">
<!--
ChangeDay = new Date("may 2, 1967 23:45:00");
Day = 24*60*60*1000;
document.write(ChangeDay);
document.write("<br>の31日後は<br>");
ChangeDay.setTime(ChangeDay.getTime() + Day * 31);
document.write(ChangeDay);
//-->
</script>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 年・月・日・時・分・秒を設定する

| | |
|---|---|
| オブジェクト名 = new Date("月 日, 年 時:分:秒") | |
| オブジェクト名.setYear() | [メソッド] |
| オブジェクト名.setMonth() | [メソッド] |
| オブジェクト名.setDate() | [メソッド] |
| オブジェクト名.setHours() | [メソッド] |
| オブジェクト名.setMinutes() | [メソッド] |
| オブジェクト名.setSeconds() | [メソッド] |




「setYear()」メソッドは年の下2桁を設定し、「setMonth()」メソッドは0を1月とした数値で月を設定し、「setDate()」メソッドは日を設定し、「setHours()」メソッドは時間を設定し、「setMinutes()」メソッドは分を設定し、「setSeconds()」メソッドは秒を設定します。サンプルでは、各メソッドを使って1度設定された年・月・日・時・分・秒の設定を変更しています。設定変更が、その部分以外の時間の要素にも影響を与えている(たとえば、秒に60以上の数値を設定した場合、分などもそれに合わせて変更されている)点に注目してください。

## Sample

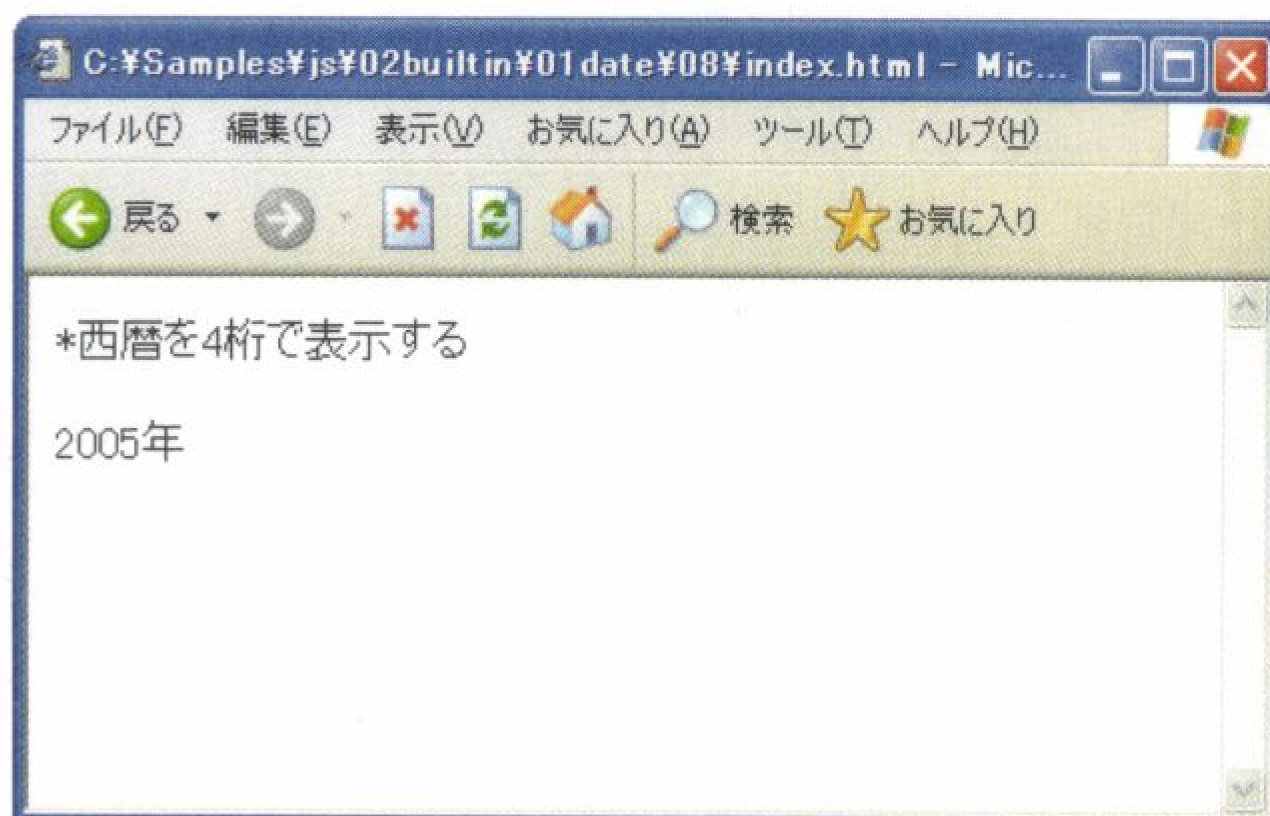
```
<script type="text/javascript">
<!--
NewDay1 = new Date("may 2, 1997 23:00:00");
document.write("設定前:",NewDay1);
document.write("<br>");
NewDay1.setYear(70);
document.write("年の設定変更:",NewDay1);
//-->
</script>
</p>
<p>
<script type="text/javascript">
<!--
NewDay2 = new Date("may 2, 1997 23:00:00");
document.write("設定前:",NewDay2);
```

日付・時間情報を利用する

```
document.write("<br>");
NewDay2.setMonth(8);
document.write("月の設定変更：",NewDay2);
//-->
</script>
</p>
<p>
<script type="text/javascript">
<!--
NewDay3 = new Date("may 2, 1997 23:00:00");
document.write("設定前：",NewDay3);
document.write("<br>");
NewDay3.setDate(34);
document.write("日の設定変更：",NewDay3);
//-->
</script>
</p>
<p>
<script type="text/javascript">
<!--
NewDay4 = new Date("may 2, 1997 23:00:00");
document.write("設定前：",NewDay4);
document.write("<br>");
NewDay4.setHours(14);
document.write("時間の設定変更：",NewDay4);
//-->
</script>
</p>
<p>
<script type="text/javascript">
<!--
NewDay5 = new Date("may 2, 1997 23:00:00");
document.write("設定前：",NewDay5);
document.write("<br>");
NewDay5.setMinutes(186);
document.write("分の設定変更：",NewDay5);
//-->
</script>
</p>
<p>
<script type="text/javascript">
<!--
NewDay6 = new Date("may 2, 1997 23:00:00");
document.write("設定前：",NewDay6);
document.write("<br>");
NewDay6.setSeconds(246);
document.write("秒の設定変更：",NewDay6);
//-->
</script>
```

# 西暦を4桁で表示する

オブジェクト名= new Date()
オブジェクト名.getFullYear() [メソッド]




「getFullYear()」メソッドは、年の値を4桁で返します。
以前からあった「getYear()」メソッドは基本的に西暦の下2桁の値しか返しませんが、「setFullYear()」メソッドの場合は、たとえば「1999」といったように、4桁すべての値を返します。
「getYear()」メソッドでも西暦2000年以降の年の値を取り扱えますが、ECMAScriptと仕様を合わせるために追加されました。
JavaScript1.3で追加されたメソッドですが、Netscape Navigator 4.0やInternet Explorer 4.Xでも使用することができます。

## Sample

```
<script type="text/javascript">
<!--
now = new Date();
document.write(now.getFullYear(),"年");
//-->
</script>
```

# 4桁の西暦を設定する

**オブジェクト名=new Date()**
**オブジェクト名.setFullYear()** [メソッド]






「setFullYear()」メソッドは、4桁の年の値を設定します。
以前からあった「setYear()」メソッドは基本的に年の下2桁をのみを設定しますが、「setFullYear()」メソッドは、サンプルのように年の値すべてを設定します。
「setYear()」メソッドでも西暦2000年以降の年の値を取り扱えますが、ECMAScriptと仕様を合わせるために追加されました。
JavaScript1.3で追加されたメソッドですが、Netscape Navigator 4.0やInternet Explorer 4.Xでも使用することができます。

## Sample

```
<script type="text/javascript">
<!--
now = new Date();
document.write("西暦設定前：",now.getFullYear(),"年");
document.write("<br>");
now.setFullYear(1999);
document.write("西暦設定後：",now.getFullYear(),"年");
//-->
</script>
```

# ミリセコンドを表示する

オブジェクト名 = new Date()
オブジェクト名.getMilliseconds()　[メソッド]




「getMilliseconds()」メソッドは、1000分の1秒の値を0から999の数値で取得します。ECMAScriptと仕様を合わせるためにJavaScript1.3で追加されたメソッドですが、Netscape Navigator 4.0やInternet Explorer 4.Xでも使用することができます。

## Sample

```
<script type="text/javascript">
<!--
now = new Date();
document.write("1000分の",now.getMilliseconds(),"秒");
//-->
</script>
```

# ミリセコンドを設定する

オブジェクト名 = new Date()
オブジェクト名.setMilliseconds()　　[メソッド]






「setMilliseconds()」メソッドは、1000分の1秒の値を0から999の数値で設定します。ECMAScriptと仕様を合わせるためにJavaScript1.3で追加されたメソッドですが、Netscape Navigator 4.0やInternet Explorer 4.Xでも使用することができます。

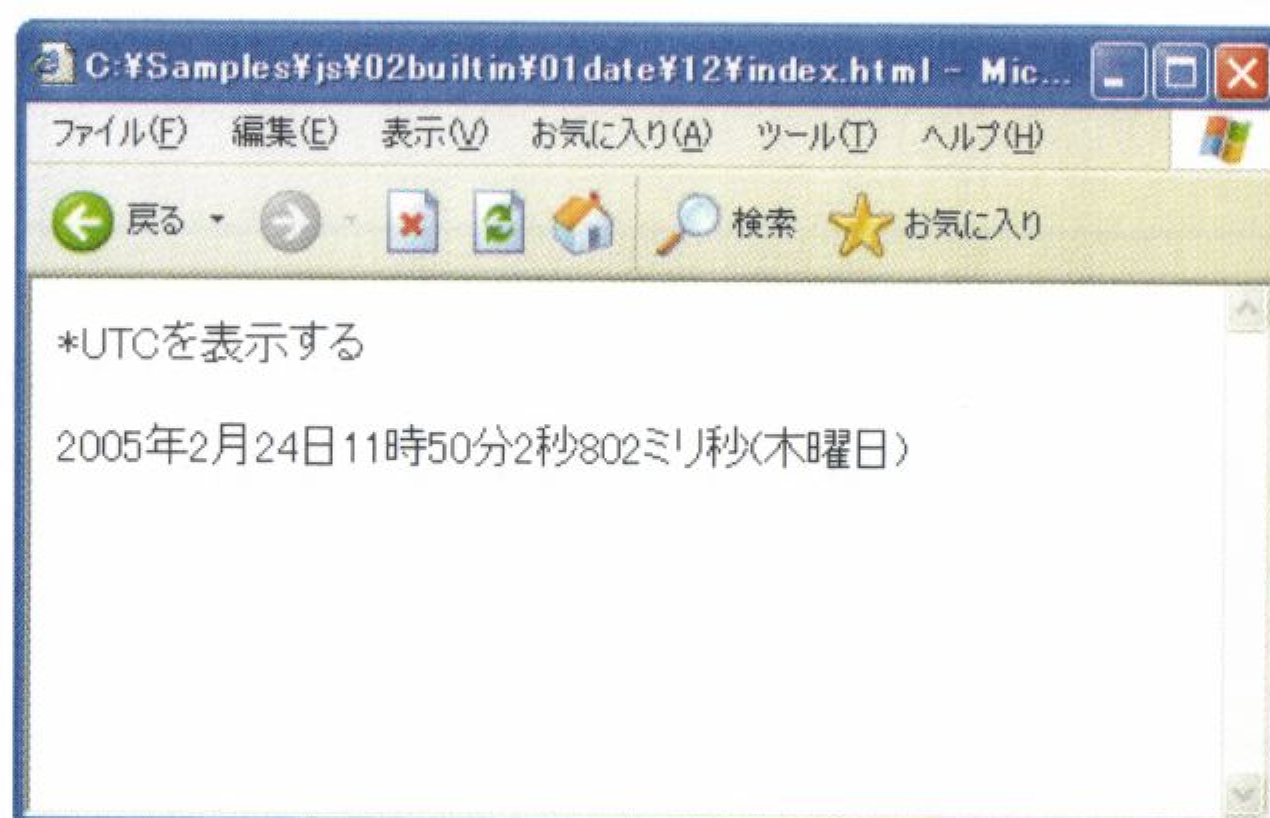
## Sample

```
<script type="text/javascript">
<!--
now = new Date();
document.write("ミリセコンド設定前：","1000分の",now.getMilliseconds(),
"秒");
document.write("<br>");
now.setMilliseconds(500);
document.write("ミリセコンド設定前：","1000分の",now.getMilliseconds(),
"秒");
//-->
</script>
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

# UTCを表示する

| | |
|---|---|
| オブジェクト名 = new Date() | |
| オブジェクト名.getUTCFullYear() | [メソッド] |
| オブジェクト名.getUTCMonth() | [メソッド] |
| オブジェクト名.getUTCDate() | [メソッド] |
| オブジェクト名.getUTCHours() | [メソッド] |
| オブジェクト名.getUTCMinutes() | [メソッド] |
| オブジェクト名.getUTCSeconds() | [メソッド] |
| オブジェクト名.getUTCMilliseconds() | [メソッド] |
| オブジェクト名.getUTCDay() | [メソッド] |




JavaScript1.3では、UTC(Coordinated Universal Time :協定世界時)を取得する多くのメソッドが追加されています。

サンプルでは、「now = new Date()」でマシンのシステム時計から現在時刻の要素を取り出したオブジェクト「now」を作成し、そこから各UTCの時間の要素を、メソッドを使って取得しています。

「getUTCFullYear()」メソッドは4桁の西暦の数値を、「getUTCMonth()」メソッドは1月を0とした月に応じた0から11までの数値を、「getUTCDate()」メソッドは日に応じた1から31までの数値を、「getUTCHours()」メソッドは時間に対応した0から23までの数値を、「getUTCMinutes()」メソッドは分に対応した0から59までの数値を、「getUTCSeconds()」メソッドは秒に対応した0から59までの数値を、「getUTCMilliseconds()」メソッドは1000分の1秒に対応した0から999までの数値を、「getDay()」メソッドは日曜日を0、月曜日を1、という順番で0から6までの数値を、UTCで取得します。

これらのメソッドは、ECMAScriptと仕様を合わせるためにJavaScript1.3で追加されたメソッドですが、Netscape Navigator 4.0やInternet Explorer 4.Xでも使用することができます。

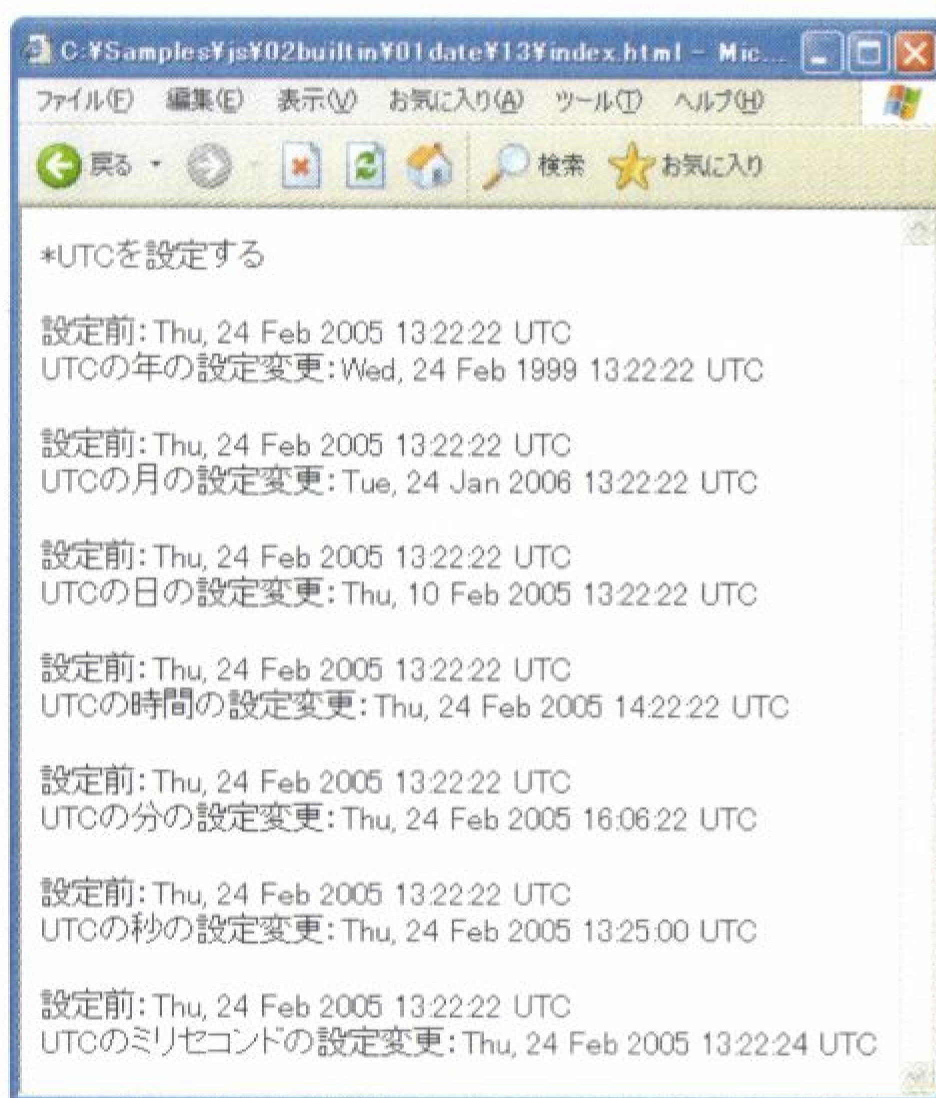
## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
var y0 = "日";
var y1 = "月";
var y2 = "火";
var y3 = "水";
var y4 = "木";
var y5 = "金";
var y6 = "土";
function getUTC(y){
    if (y==0) { document.write( y0.fontcolor("red") ) }
    if (y==1) { document.write( y1 ) }
    if (y==2) { document.write( y2 ) }
    if (y==3) { document.write( y3 ) }
    if (y==4) { document.write( y4 ) }
    if (y==5) { document.write( y5 ) }
    if (y==6) { document.write( y6.fontcolor("blue") ) }
}
//-->
</script>
  ～中略～
</head>
<body>
*UTCを表示する
<p>
<script type="text/javascript">
<!--
now = new Date();
    document.write(now.getUTCFullYear(),"年");
    document.write(now.getUTCMonth()+1,"月",now.getUTCDate(),"日");
    document.write(now.getUTCHours(),"時",now.getUTCMinutes(),"分");
    document.write(now.getUTCSeconds(),"秒",now.getUTCMilliseconds
(),"ミリ秒");
    document.write("(");
    getUTC(now.getUTCDay());
    document.write("曜日)");
//-->
</script>
</p>
</body>
</html>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# UTCを設定する

| | |
|---|---|
| オブジェクト名 = new Date() | |
| オブジェクト名.toUTCString() | [メソッド] |
| オブジェクト名.setUTCFullYear() | [メソッド] |
| オブジェクト名.setUTCMonth() | [メソッド] |
| オブジェクト名.setUTCDate() | [メソッド] |
| オブジェクト名.setUTCHours() | [メソッド] |
| オブジェクト名.setUTCMinutes() | [メソッド] |
| オブジェクト名.setUTCSeconds() | [メソッド] |
| オブジェクト名.setUTCMilliseconds() | [メソッド] |




JavaScript1.3では、UTC(Coordinated Universal Time :協定世界時)を設定する多くのメソッドが追加されています。
サンプルでは、ページが読み込まれた時の時間の値を、各メソッドを使って変更しています。
「setUTCFullYear()」メソッドは4桁のUTCの年の値を設定し、「setUTCMonth()」メソッドは0を1月とした数値でUTCの月の値を設定し、「setUTCDate()」メソッドはUTCの日の値を設定し、「setUTCHours()」メソッドはUTCの時間の値を設定し、「setUTCMinutes()」メソッドはUTCの分の値を設定し、「setUTCSeconds()」メソッドはUTCの秒の値を設定します。なお、「toUTCString()」メソッドは、UTCの日付と時間を文字列へ変換します。
これらのメソッドは、ECMAScriptと仕様を合わせるためにJavaScript1.3で追加されたメソッドですが、Netscape Navigator 4.0やInternet Explorer 4.Xでも使用することができます。

**Sample**

```
<p>
<script type="text/javascript">
<!--
NewDay1 = new Date();
document.write("設定前:",NewDay1.toUTCString());
document.write("<br>");
NewDay1.setUTCFullYear(1999);
document.write("UTCの年の設定変更:",NewDay1.toUTCString());
//-->
</script>
</p>
<p>
<script type="text/javascript">
<!--
NewDay2 = new Date();
document.write("設定前:",NewDay2.toUTCString());
document.write("<br>");
NewDay2.setUTCMonth(12);
document.write("UTCの月の設定変更:",NewDay2.toUTCString());
//-->
</script>
</p>
<p>
<script type="text/javascript">
<!--
NewDay3 = new Date();
document.write("設定前:",NewDay3.toUTCString());
document.write("<br>");
NewDay3.setUTCDate(10);
document.write("UTCの日の設定変更:",NewDay3.toUTCString());
//-->
</script>
</p>
<p>
<script type="text/javascript">
<!--
NewDay4 = new Date();
document.write("設定前:",NewDay4.toUTCString());
document.write("<br>");
NewDay4.setUTCHours(14);
document.write("UTCの時間の設定変更:",NewDay4.toUTCString());
//-->
</script>
</p>
```

```
<p>
<script language="JavaScript">
<!--
NewDay5 = new Date();
document.write("設定前：",NewDay5.toUTCString());
document.write("<br>");
NewDay5.setUTCMinutes(186);
document.write("UTCの分の設定変更：",NewDay5.toUTCString());
//-->
</script>
</p>
<p>
<script language="JavaScript">
<!--
NewDay6 = new Date();
document.write("設定前：",NewDay6.toUTCString());
document.write("<br>");
NewDay6.setUTCSeconds(180);
document.write("UTCの秒の設定変更：",NewDay6.toUTCString());
//-->
</script>
</p>
<p>
<script language="JavaScript">
<!--
NewDay7 = new Date();
document.write("設定前：",NewDay7.toUTCString());
document.write("<br>");
NewDay7.setUTCMilliseconds(2000);
document.write("UTCのミリセコンドの設定変更：",NewDay7.toUTCString());
//-->
</script>
</p>
```

# 日付をカウントダウンする

**オブジェクト名 = new Date()**
**オブジェクト名 = new Date(年, 月, 日)**
**オブジェクト名.getTime()** [メソッド]






サンプルではまず、「today = new Date()」で現在の時間の要素を持った「today」というオブジェクトと、「XDay = new Date(2010,0,1)」で西暦2010年1月1日の要素を持った「XDay」というオブジェクトを作っています。そして、「getTime()」メソッドを使って、それぞれのオブジェクトの1970年1月1日0時0分0秒からのミリ秒単位の経過時間を取得し、その差の値を「/(24*60*60*1000)」とすることによって、日数に戻しています。「new Date(2010,0,1)」内の日付を変更することによって、希望の日付までのカウントダウンをすることができます。

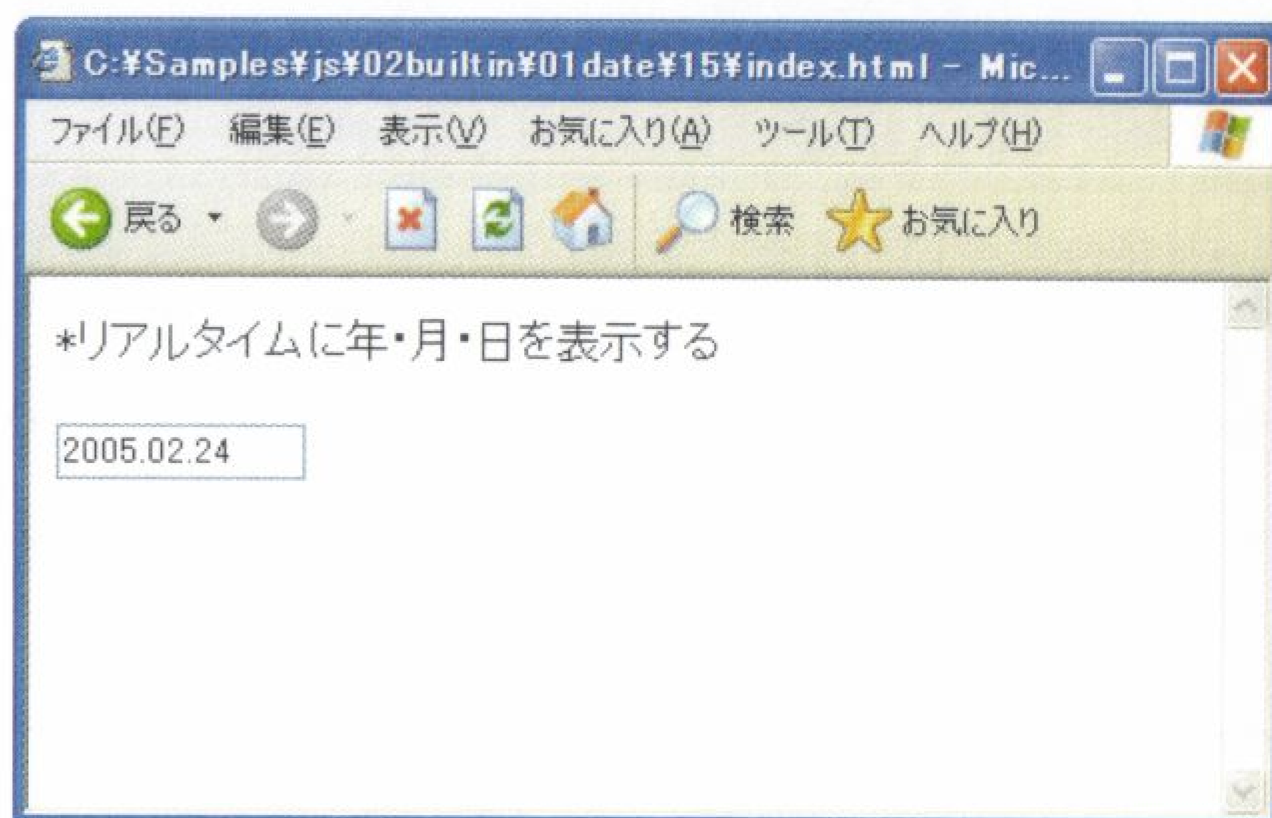
## Sample

```
<script type="text/javascript">
<!--
today = new Date();
XDay = new Date(2010,0,1);
NC = (XDay.getTime()-today.getTime())/(24*60*60*1000);
document.write("西暦2010年まであと"+ Math.ceil(NC) +"日");
//-->
</script>
```


Math.ceil()：「Mathオブジェクト」の「もっとも近くて大きい整数を返す」(P.536)

# リアルタイムに年・月・日を表示する

オブジェクト名 = new Date()
オブジェクト名.getYear() [メソッド]
オブジェクト名.getMonth() [メソッド]
オブジェクト名.getDate() [メソッド]




サンプルでは、Dateオブジェクトの「getYear()」メソッドで年を、「getMonth()」メソッドで月を、「getDate()」メソッドで日を取得し、フォーム内に書き出す処理を0.5秒ごとに繰り返しています。
フォームに書き出す時に、年は西暦の下2桁を取得するので4桁の年号に直す処理を、月は取得した数値に1を加えて10以下の時には頭に0を付ける処理を、日は取得した数値が10以下の時には頭に0を付ける処理をしています。
また、Netscape Navigator 2.0のメモリーエラー対策として、10分後にスクリプトを停止する処理をしています。もし不要な場合は、「//10分後にタイマーを止める処理」のコメントがある部分の1行を削除してください。
「getYear()」メソッドで取得した値を4桁の年号に直す方法に関しては、付録コラムの「Javascriptの2000年問題」(P.614)を参照してください。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
```

```
<script type="text/javascript">
<!--
var TC = 0;
function DayWatch() {
    if (TC < 1200) {     //10分後にタイマーを止める処理
    TC++ ;               //10分後にタイマーを止める処理
    var day = new Date();
    if ( day.getYear() >= 2000 ){ var year = day.getYear() }
    else { var year = day.getYear() +1900 }
    var month = day.getMonth()+1;
    var date = day.getDate();
        if (month < 10) {     //月.日が1桁の時は頭に0を付ける処理
            month = "0" + month;
                            }
        if (date < 10) {
            date = "0" + date;
                           }
    document.Watch1.watch1.value = year +"."+month+"."+date;
    setTimeout("DayWatch()", 500);
        }                    //10分後にタイマーを止める処理
}
//-->
</script>

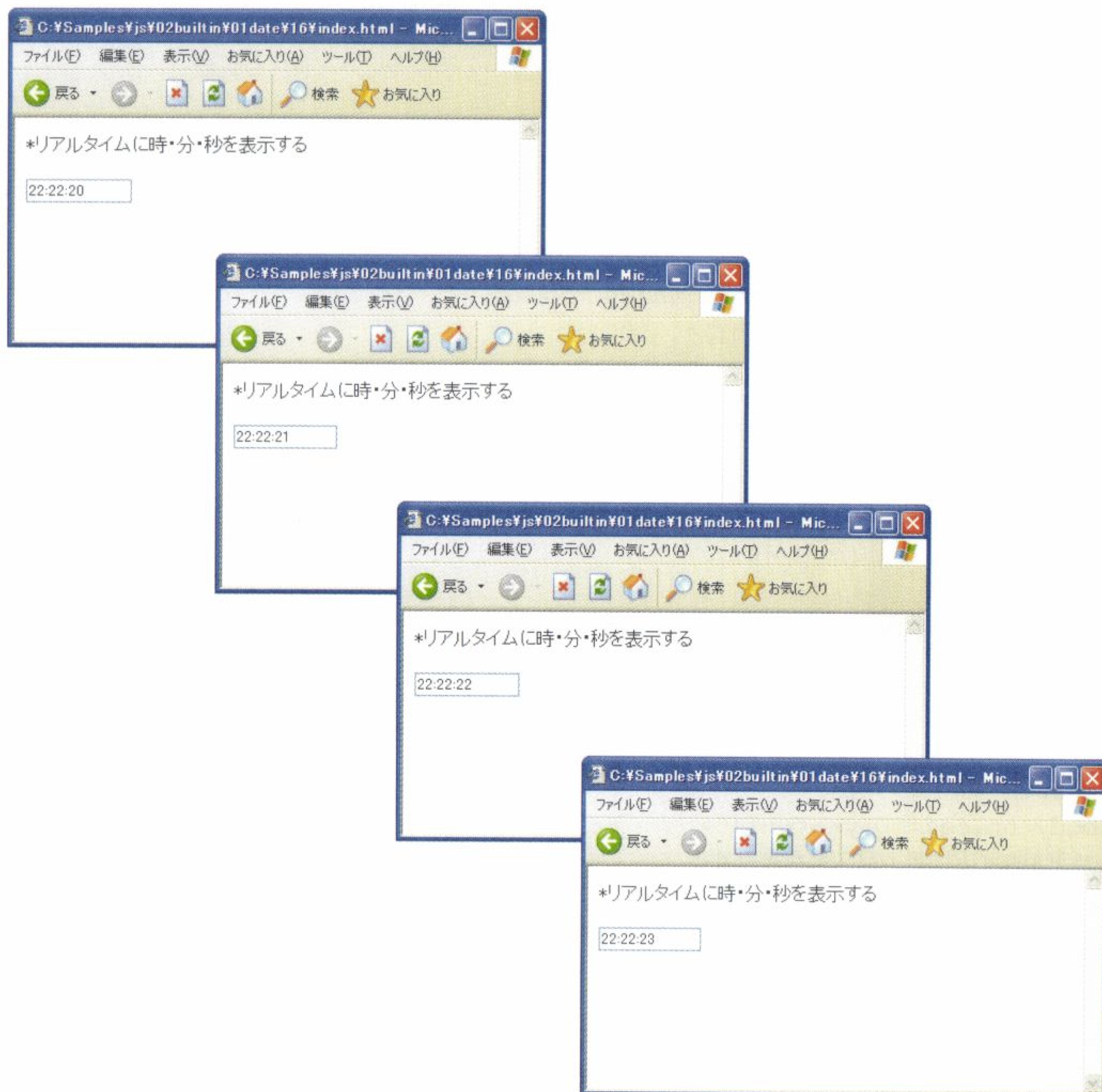
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body onLoad="DayWatch()">
*リアルタイムに年・月・日を表示する
<p>
<form name="Watch1">
<input type="text" name="watch1" size="15">
</form>
</p>
</body>
</html>
```

setTimeout()・clearTimeout()：「windowオブジェクト」の「ステータス行に文字を流す」(P.364)
<input type="text">：「formオブジェクト」の「フォームに文字を流す」(P.430)

# リアルタイムに時・分・秒を表示する

オブジェクト名 = new Date()
オブジェクト名.getHours() [メソッド]
オブジェクト名.getMinutes() [メソッド]
オブジェクト名.getSeconds() [メソッド]




サンプルでは、Dateオブジェクトの「getHours()」メソッドで時間を、「getMinutes()」メソッドで分を、「getSeconds()」メソッドで秒を取得し、フォーム内に書き出す処理を0.5秒ごとに繰り返しています。

フォームに値を書き出す時に取得した各数値が10以下の場合、頭に0を付ける処理をしています。

また、Netscape Navigator 2.0のメモリーエラー対策として、10分後にスクリプトを停止する処理をしています。もし不要な場合は、「//10分後にタイマーを止める処理」のコメントがある部分の1行を削除してください。

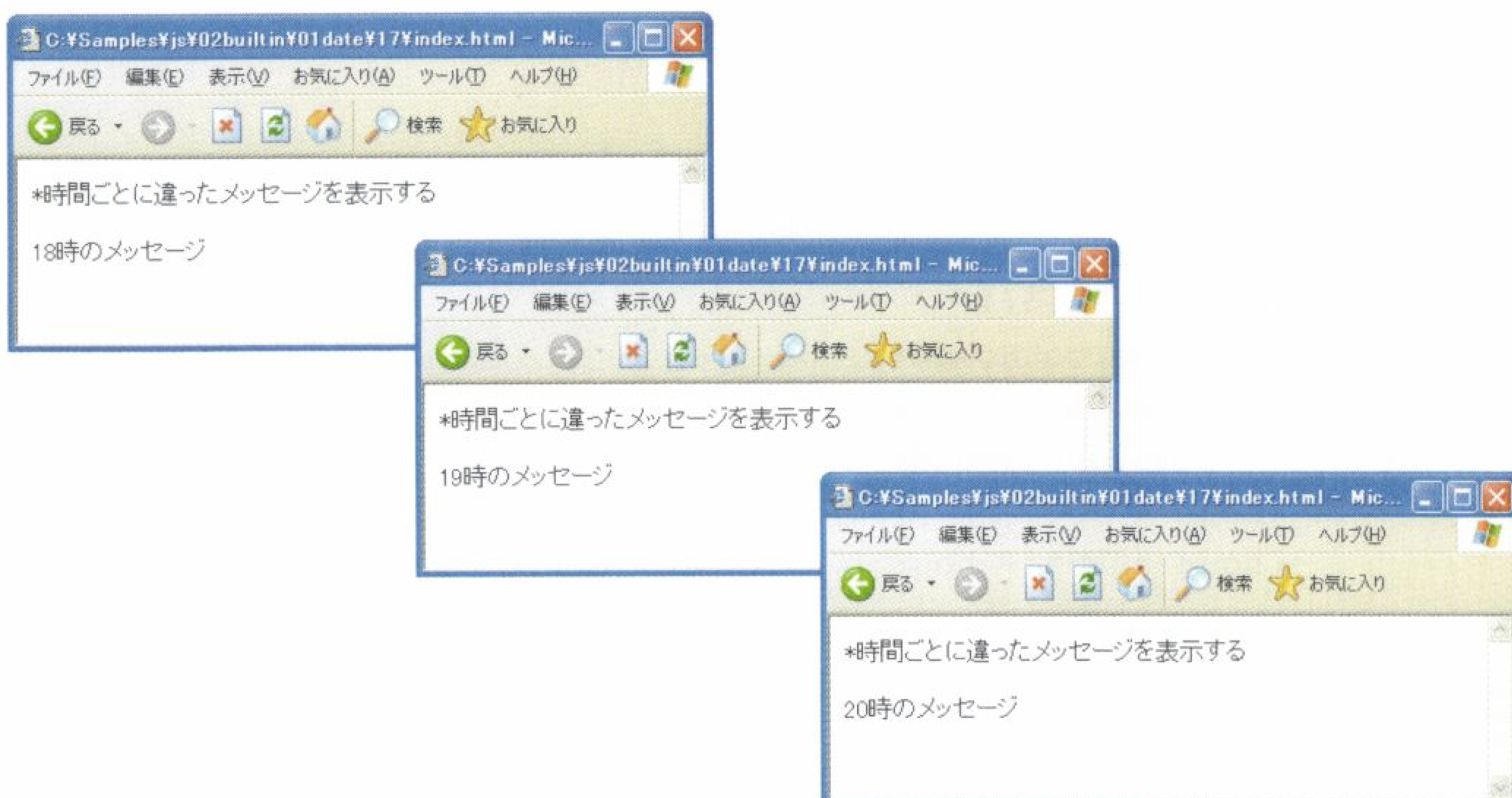
## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
var TC = 0;
function TimeWatch() {
   if (TC < 1200) {     //10分後にタイマーを止める処理
        TC++ ;          //10分後にタイマーを止める処理
    var time = new Date();
    var hour = time.getHours();
    var min = time.getMinutes();
    var sec = time.getSeconds();
        if (hour < 10) {     //時・分・秒が1桁の時は頭に0を付ける処理
            hour = "0" + hour;
                        }
        if (min < 10) {
            min = "0" + min;
                      }
        if (sec < 10) {
            sec = "0" + sec;
                      }
    document.Watch2.watch2.value = hour+':'+min+':'+sec;
    setTimeout("TimeWatch()", 500);
                  }     //10分後にタイマーを止める処理
}
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body onLoad="TimeWatch()">
*リアルタイムに時・分・秒を表示する
<p>
<form name="Watch2">
<input type="text" name="watch2" size="15">
</form>
</p>
</body></html>
```

 setTimeout()・clearTimeout()：「windowオブジェクト」の「ステータス行に文字を流す」(P.364)
<input type="text">：「formオブジェクト」の「フォームに文字を流す」(P.430)

# 時間ごとに違ったメッセージを表示する

オブジェクト名 = new Date()
オブジェクト名.getHours() ［メソッド］




サンプルでは、HTMLファイルがロードされた時に、関数「geth()」内の「h.getHours()」が時間の値を取得して、その値を「function geth(t){～}」内のif文で参照し、時間に合わせたメッセージを表示しています。
メッセージ部分に「<img src='URL' alt='xxx' width="xxx" height="xxx">」と画像を表示するタグを書けば、時間によって違った画像を表示することもできます。
サンプルのように1時間ごとにひとつのメッセージを設定している場合は、本当はif文をネスト（入れ子）にする必要はありません。数時間ごとに処理を行う方法は、「時間によって背景画像を変える」（次項）を参考にしてください。

## Sample

```
<script type="text/javascript">
<!--
var m0 = "0時のメッセージ";
var m1 = "1時のメッセージ";
var m2 = "2時のメッセージ";
var m3 = "3時のメッセージ";
var m4 = "4時のメッセージ";
var m5 = "5時のメッセージ";
var m6 = "6時のメッセージ";
var m7 = "7時のメッセージ";
var m8 = "8時のメッセージ";
var m9 = "9時のメッセージ";
var m10 = "10時のメッセージ";
```

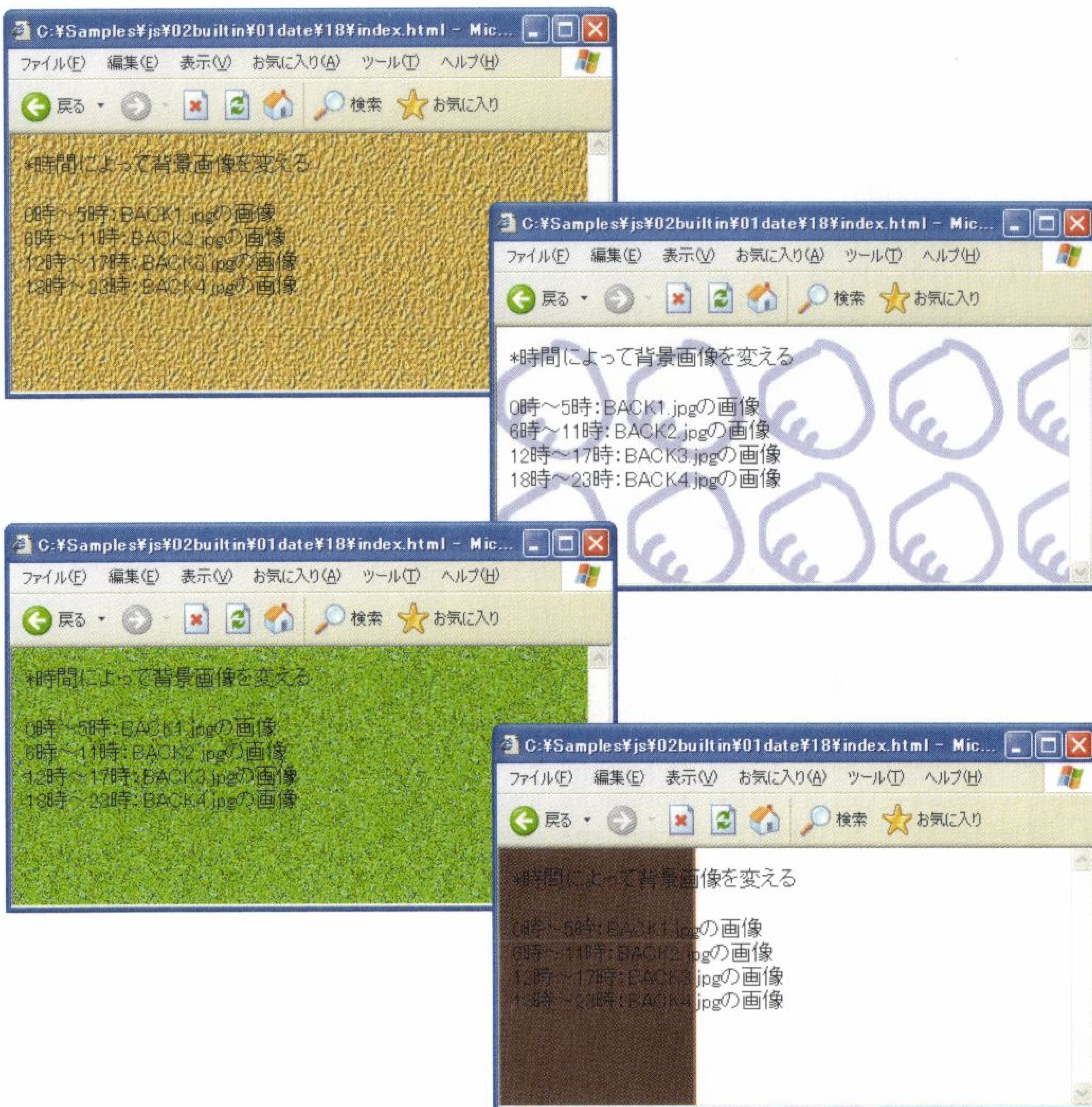
```
var m11 = "11時のメッセージ";
var m12 = "12時のメッセージ";
var m13 = "13時のメッセージ";
var m14 = "14時のメッセージ";
var m15 = "15時のメッセージ";
var m16 = "16時のメッセージ";
var m17 = "17時のメッセージ";
var m18 = "18時のメッセージ";
var m19 = "19時のメッセージ";
var m20 = "20時のメッセージ";
var m21 = "21時のメッセージ";
var m22 = "22時のメッセージ";
var m23 = "23時のメッセージ";
function geth(t){
    if (t==0) document.write( m0 );
        else { if (t==1) document.write( m1 );
        else { if (t==2) document.write( m2 );
        else { if (t==3) document.write( m3 );
        else { if (t==4) document.write( m4 );
        else { if (t==5) document.write( m5 );
        else { if (t==6) document.write( m6 );
        else { if (t==7) document.write( m7 );
        else { if (t==8) document.write( m8 );
        else { if (t==9) document.write( m9 );
        else { if (t==10) document.write( m10 );
        else { if (t==11) document.write( m11 );
        else { if (t==12) document.write( m12 );
        else { if (t==13) document.write( m13 );
        else { if (t==14) document.write( m14 );
        else { if (t==15) document.write( m15 );
        else { if (t==16) document.write( m16 );
        else { if (t==17) document.write( m17 );
        else { if (t==18) document.write( m18 );
        else { if (t==19) document.write( m19 );
        else { if (t==20) document.write( m20 );
        else { if (t==21) document.write( m21 );
        else { if (t==22) document.write( m22 );
        else { if (t==23) document.write( m23 );
    }}}}}}}}}}}}}}}}}}}}}}}
}
//-->
</script>
 ～中略～
<script type="text/javascript">
<!--
    h = new Date();
    geth(h.getHours());
//-->
</script>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

日付・時間情報を利用する

# 時間によって背景画像を変える

**オブジェクト名 = new Date()**
**オブジェクト名.getHours()** [メソッド]




Netscape 6.0では、DOMの採用によって、背景画像をJavaScriptを使って操作することが可能になりました。しかし、それ以前のバージョンのNetscape Navigatorでは、背景画像を操作することはできませんでした。
そのためサンプルでは、</head>と<body>の間に、時間に応じた背景画像の指定を含む<body>を書き出すようにしています。
<body>の中に、画像だけでなく、フォントの色などを指定しても評価されます。けれども、<body>がふたつあることになるので、HTMLの文法的にはあまり正しいとはいえません。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<title></title>

<script type="text/javascript">
<!--
function geth(t){
    if (t<=5) document.write("<body background='BACK1.jpg'>");
        else { if (t<=11) document.write("<body background='BACK2.jpg'>");
        else { if (t<=17) document.write("<body background='BACK3.jpg'>");
        else { if (t<=23) document.write("<body background='BACK4.jpg'>");
    }}}
}
//-->
</script>

</head>

<script type="text/javascript">
<!--
h = new Date();
   geth(h.getHours());
//-->
</script>

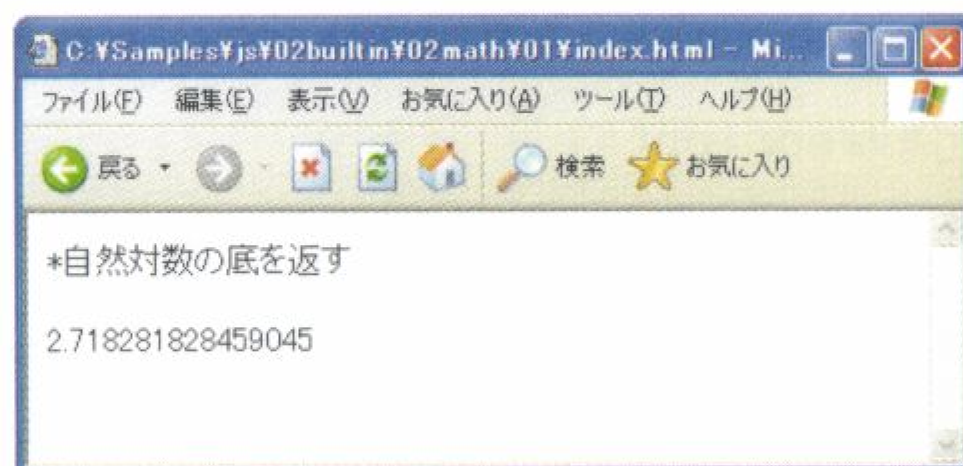
<body>
*時間によって背景画像を変える
<p>
0時～5時：BACK1.jpgの画像<br>
6時～11時：BACK2.jpgの画像<br>
12時～17時：BACK3.jpgの画像<br>
18時～23時：BACK4.jpgの画像
</p>
</body>
</html>
```


<body>：「基本的な内容」の「最低限必要な要素」(P.11)
<body background='URL'>：「基本的な内容」の「全体の背景画像を設定する」(P.14)

JavaScript

## 自然対数の底を返す

**Math.E** [プロパティ]




「E」プロパティは、自然対数の底(オイラー定数)の値を持っています。
このプロパティは、読み出し専用です。

### Sample

```
<script type="text/javascript">
<!--
document.write(Math.E);
//-->
</script>
```

JavaScript

## eを底とする2の自然対数を返す

**Math.LN2** [プロパティ]




「LN2」プロパティは、e(オイラー定数)を底とする2の自然対数を持っています。
このプロパティは、読み出し専用です。

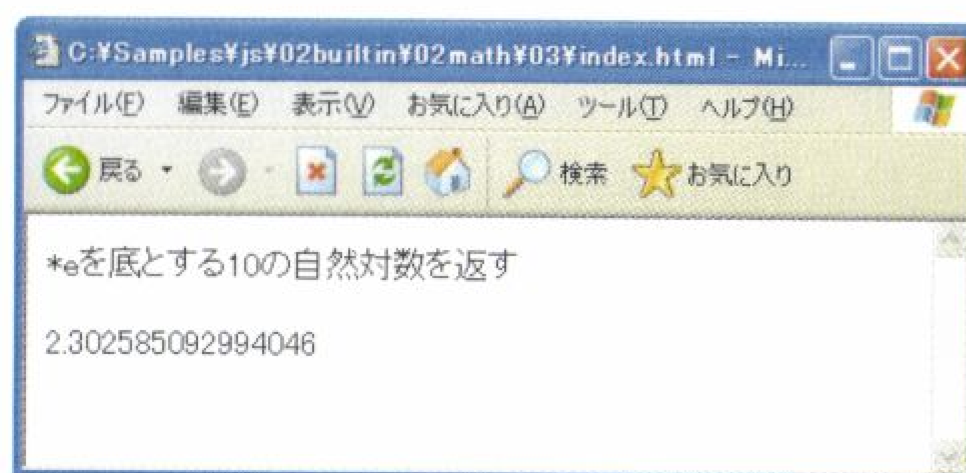
### Sample

```
<script type="text/javascript">
<!--
document.write(Math.LN2);
//-->
</script>
```

JavaScript

# eを底とする10の自然対数を返す

**Math.LN10** [プロパティ]




「LN10」プロパティは、e(オイラー定数)を底とする10の自然対数を持っています。このプロパティは、読み出し専用です。

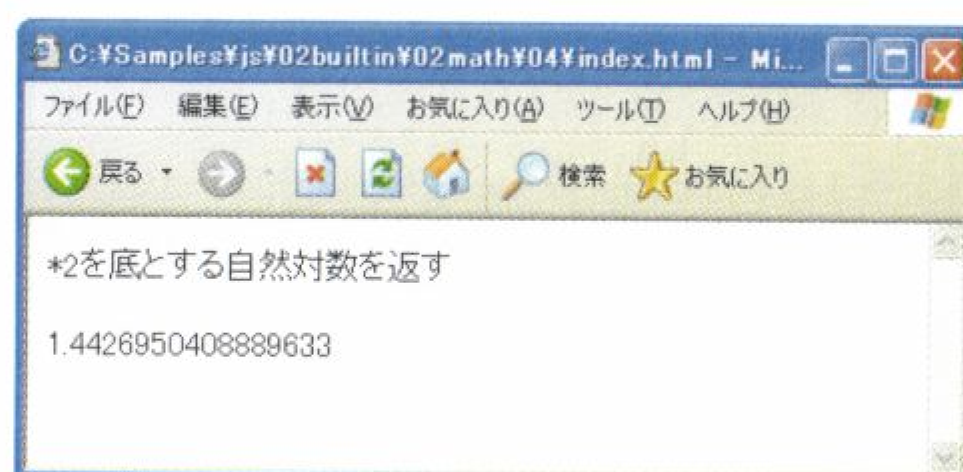
## Sample

```
<script type="text/javascript">
<!--
document.write(Math.LN10);
//-->
</script>
```

JavaScript

# 2を底とする自然対数を返す

**Math.LOG2E** [プロパティ]




「LOG2E」プロパティは、2を底とする自然対数を持っています。このプロパティは、読み出し専用です。

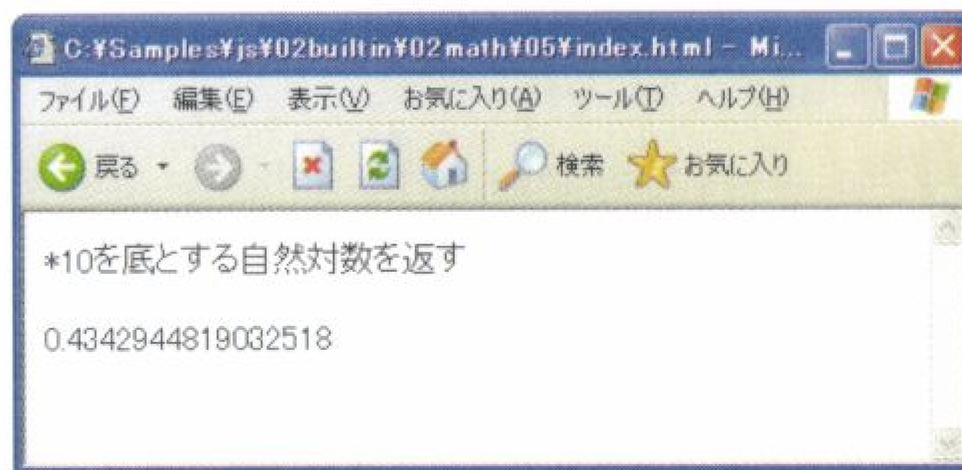
## Sample

```
<script type="text/javascript">
<!--
document.write(Math.LOG2E);
//-->
</script>
```

JavaScript

# 10を底とする自然対数を返す

**Math.LOG10E** ［プロパティ］




「LOG10E」プロパティは、10を底とする自然対数を持っています。
このプロパティは、読み出し専用です。

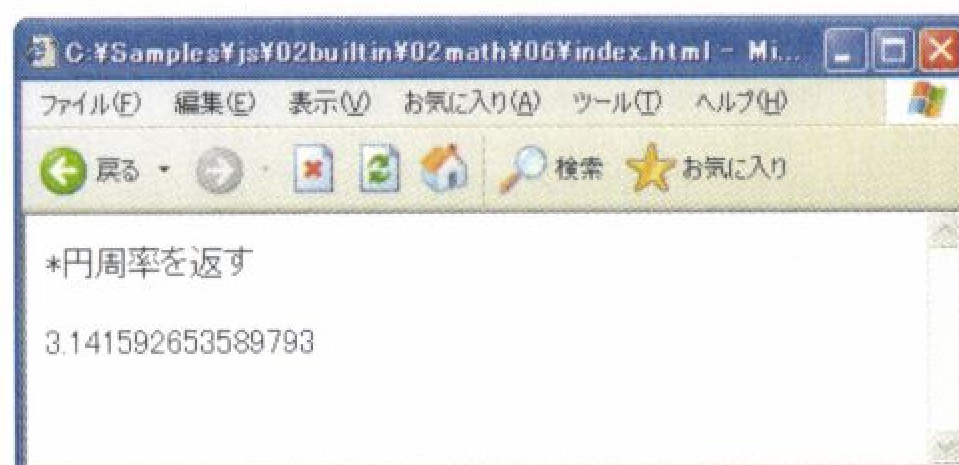
## Sample

```
<script type="text/javascript">
<!--
document.write(Math.LOG10E);
//-->
</script>
```

JavaScript

# 円周率を返す

**Math.PI** ［プロパティ］




「PI」プロパティは、円周率を持っています。
このプロパティは、読み出し専用です。

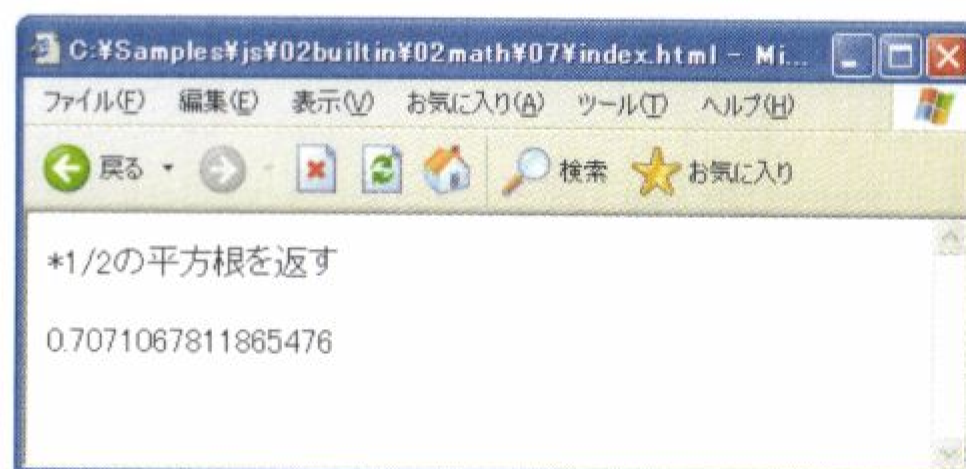
## Sample

```
<script type="text/javascript">
<!--
document.write(Math.PI);
//-->
</script>
```

JavaScript

# 1/2の平方根を返す

**Math.SQRT1_2** [プロパティ]




「SQRT1_2」プロパティは、ルート2分の1の値を持っています。
このプロパティは、読み出し専用です。

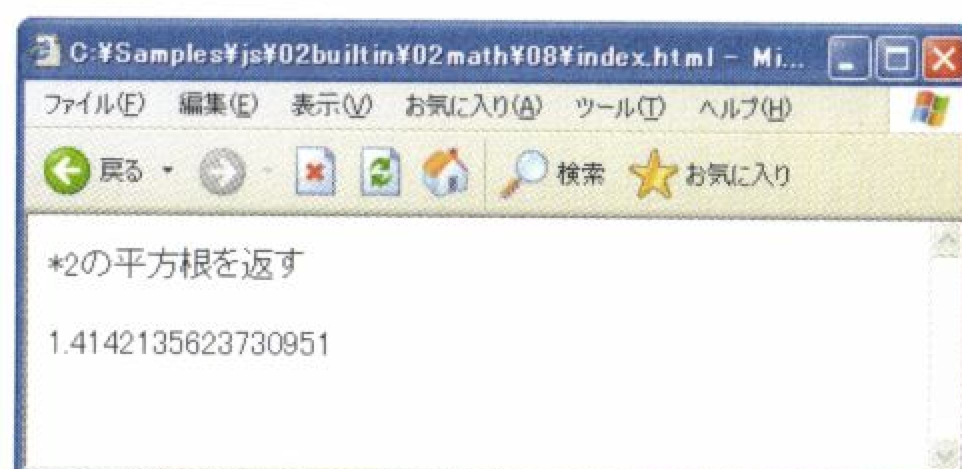
## Sample

```
<script type="text/javascript">
<!--
document.write(Math.SQRT1_2);
//-->
</script>
```

JavaScript

# 2の平方根を返す

**Math.SQRT2** [プロパティ]




「SQRT2」プロパティは、ルート2の値を持っています。
このプロパティは、読み出し専用です。

## Sample

```
<script type="text/javascript">
<!--
document.write(Math.SQRT2);
//-->
</script>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 0からの絶対値を返す

**Math.abs(*n*)** [メソッド]




「abs()」メソッドは、nの0からの絶対値（0からの距離）を返します。
サンプルの通り、「-5」も「5」も、「5」となります。

## Sample

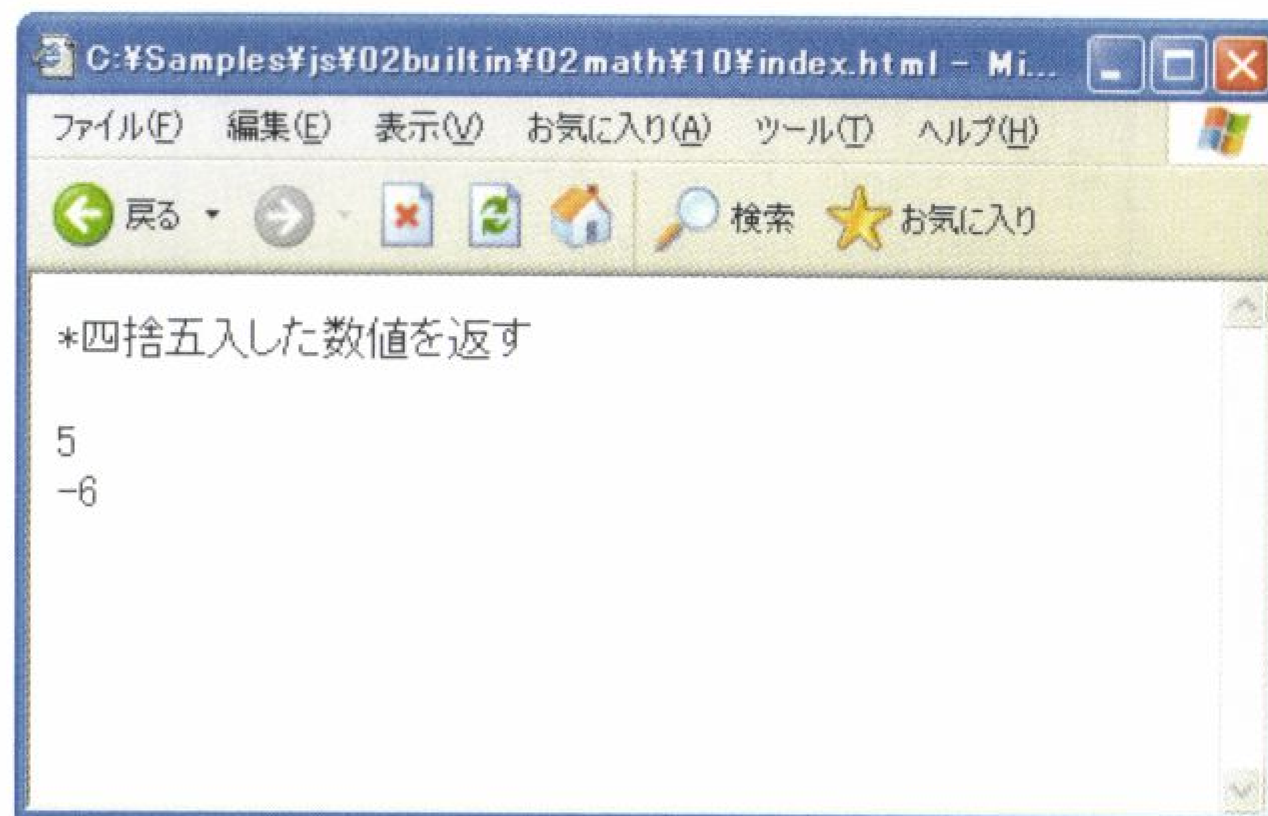
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*0からの絶対値を返す
<p>
<script type="text/javascript">
<!--
document.write(Math.abs(-5));
document.write("<br>");
document.write(Math.abs(5));
//-->
</script>
</p>
</body>
</html>
```

演算する

# 四捨五入した数値を返す

**Math.round(*n*)** [メソッド]






「round()」メソッドは、nの小数第1位を四捨五入した数値を返します。

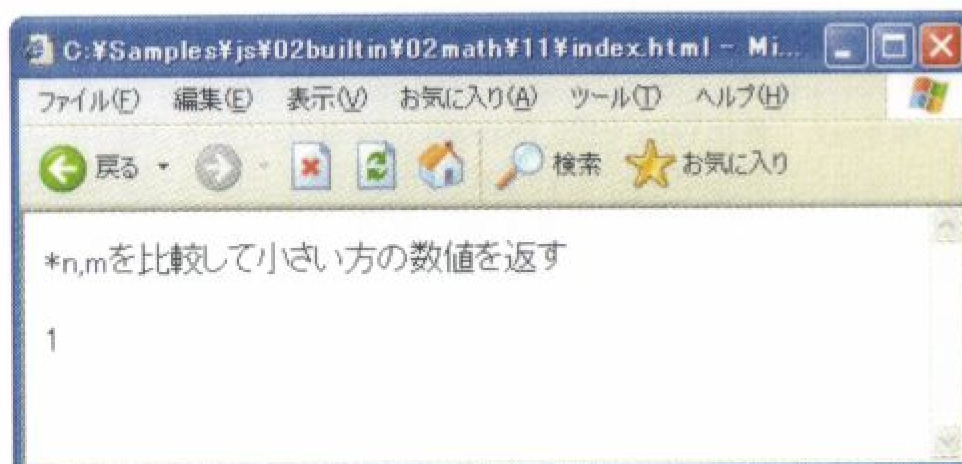
## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*四捨五入した数値を返す
<p>
<script type="text/javascript">
<!--
document.write(Math.round(5.49));
document.write("<br>");
document.write(Math.round(-5.64));
//-->
</script>
</p>
</body>
</html>
```

JavaScript

## n,m を比較して小さい方の数値を返す

**Math.min(*n*,*m*)** ［メソッド］




「min()」メソッドは、n と m を比較して小さい方の数値を返します。
「max()」メソッドと逆の働きをします。

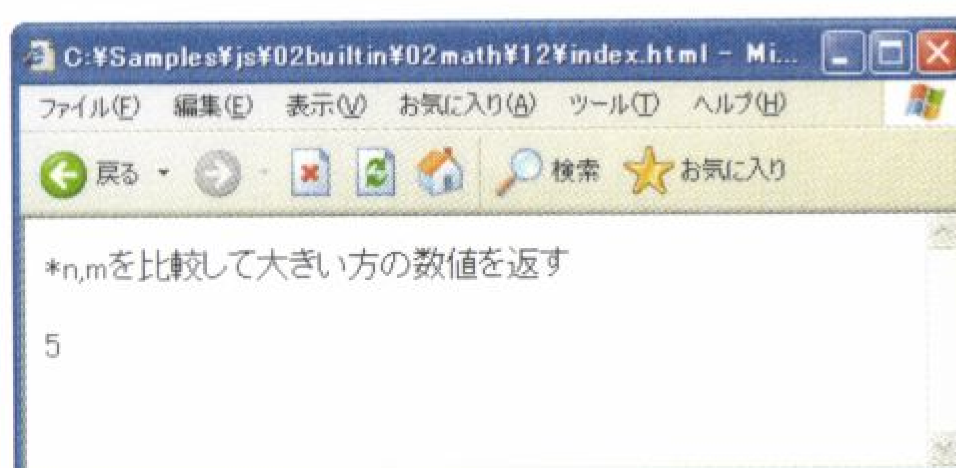
**Sample**

```
<script type="text/javascript">
<!--
document.write(Math.min(1,5));
//-->
</script>
```

JavaScript

## n,m を比較して大きい方の数値を返す

**Math.max(*n*, *m*)** ［メソッド］




「max()」メソッドは、n と m を比較して大きい方の数値を返します。
「min()」メソッドと逆の働きをします。

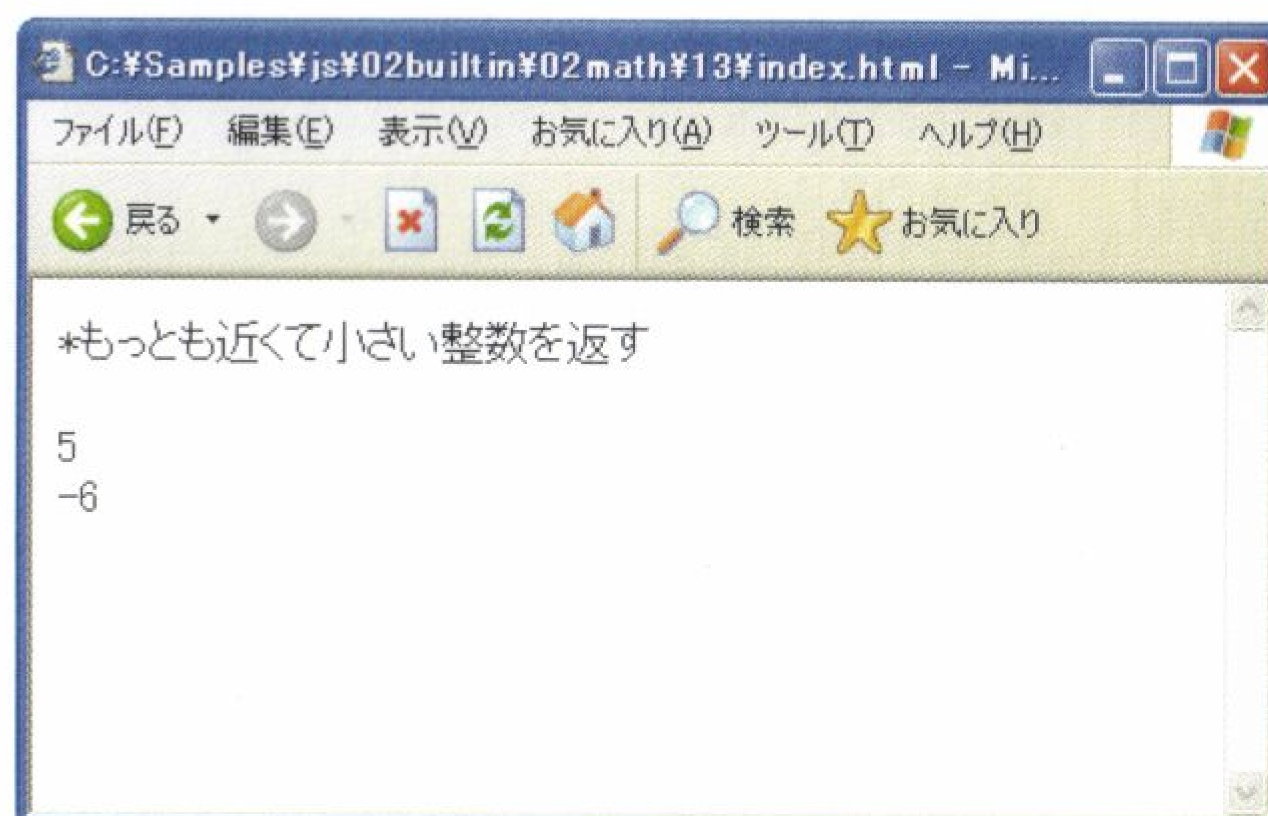
**Sample**

```
<script type="text/javascript">
<!--
document.write(Math.max(1,5));
//-->
</script>
```

# もっとも近くて小さい整数を返す

**Math.floor(*n*)** [メソッド]





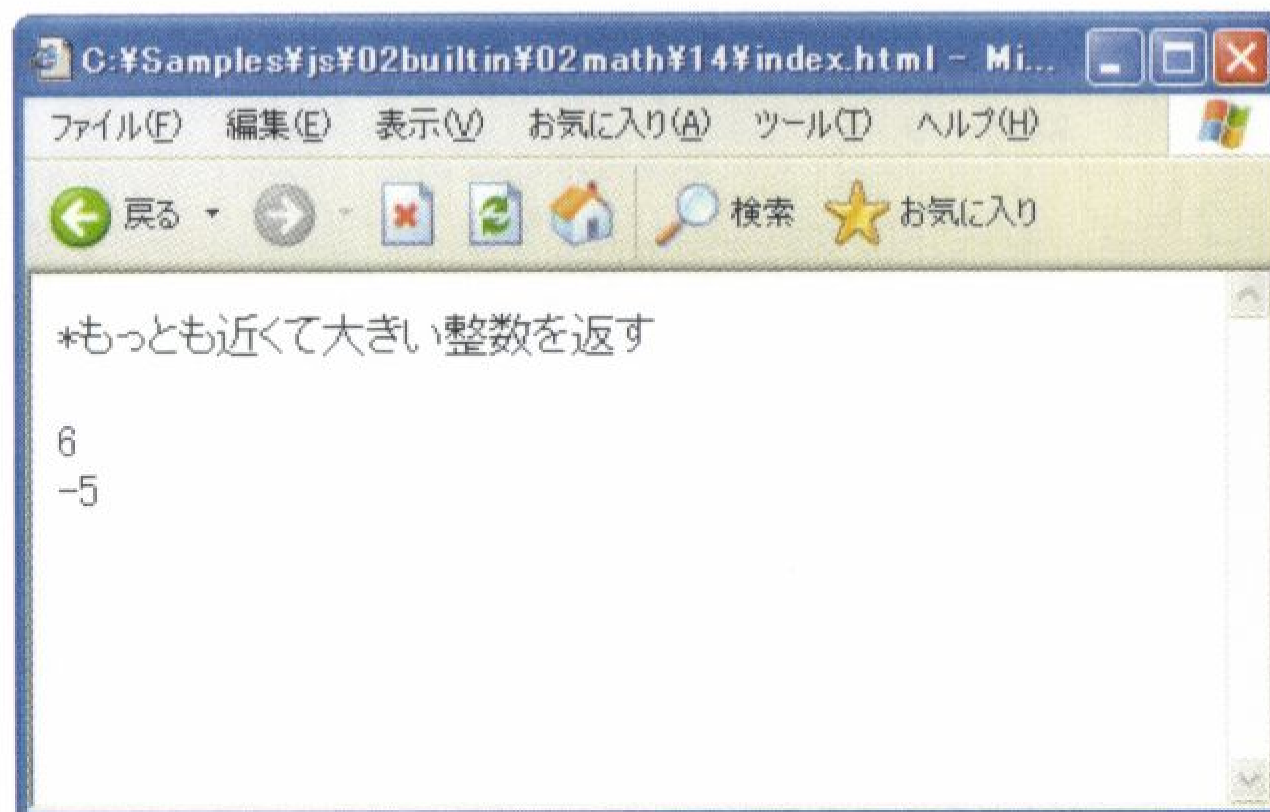

「floor()」メソッドは、nにもっとも近くて小さい整数を返します。
「ceil()」メソッドと逆の働きをします。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*もっとも近くて小さい整数を返す
<p>
<script type="text/javascript">
<!--
document.write(Math.floor(5.2));
document.write("<br>");
document.write(Math.floor(-5.2));
//-->
</script>
</p>
</body>
</html>
```

# もっとも近くて大きい整数を返す

**Math.ceil(*n*)** [メソッド]



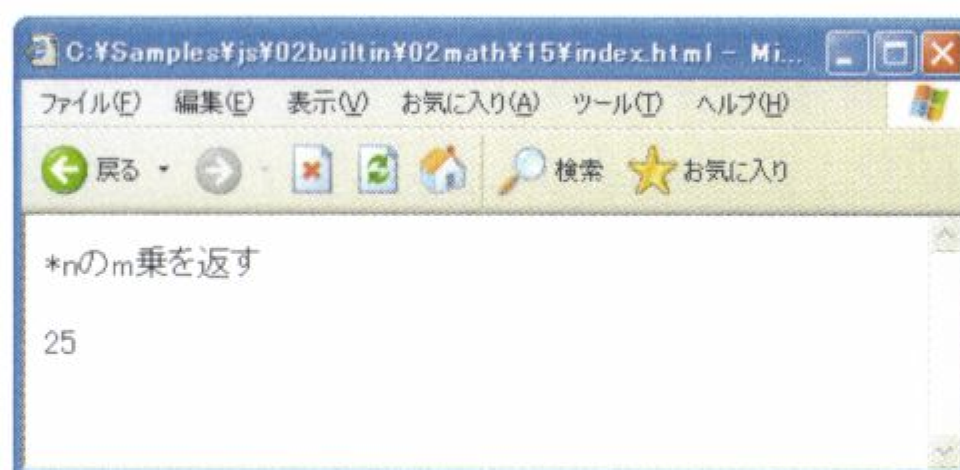

「ceil()」メソッドは、nにもっとも近くて大きい整数を返します。
「floor()」メソッドと逆の働きをします。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*もっとも近くて大きい整数を返す
<p>
<script type="text/javascript">
<!--
document.write(Math.ceil(5.2));
document.write("<br>");
document.write(Math.ceil(-5.2));
//-->
</script>
</p>
</body>
</html>
```

# nのm乗を返す

**Math.pow(*n*,*m*)** [メソッド]




「pow()」メソッドは、nのm乗の数値を返します。

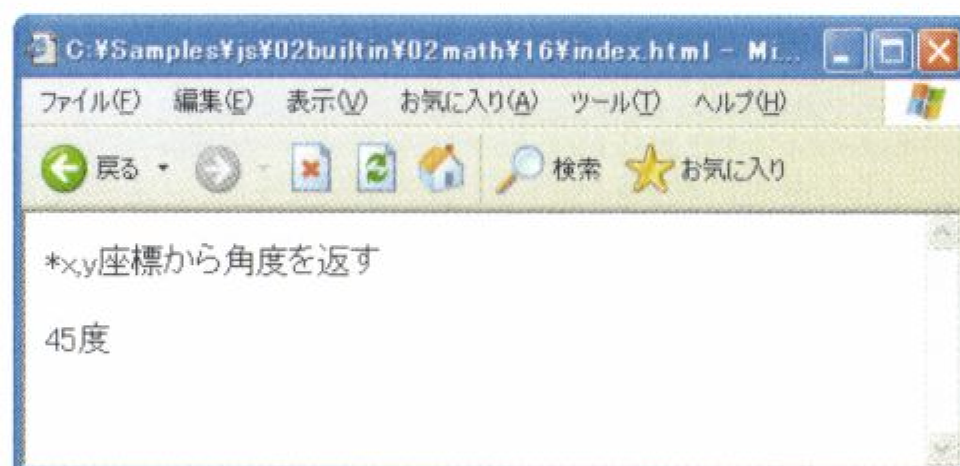
## Sample

```
<script type="text/javascript">
<!--
document.write(Math.pow(5,2));
//-->
</script>
```



# x,y座標から角度を返す

**Math.atan2(*x*,*y*)** [メソッド]




「atan2()」メソッドは、x,y座標から角度を求めます。単位はラジアンです。
サンプルでは、ラジアンから度を求めています。
ドキュメント化はJavaScript1.1からですが、JavaScript1.0でも使用できます。

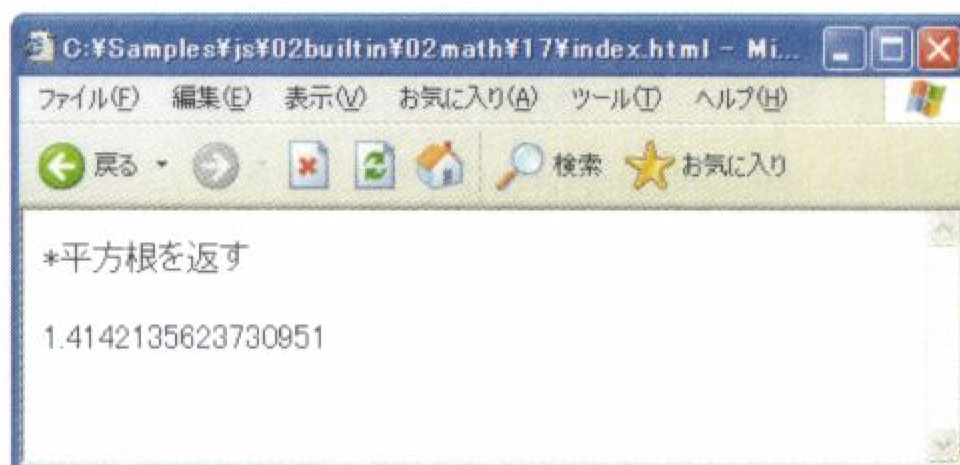
## Sample

```
<script type="text/javascript">
<!--
document.write(Math.atan2(5,5) * 180 / Math.PI,"度");
//-->
</script>
```

JavaScript

## 平方根を返す

**Math.sqrt(*n*)** [メソッド]




「sqrt()」メソッドは、nの平方根の値を返します。

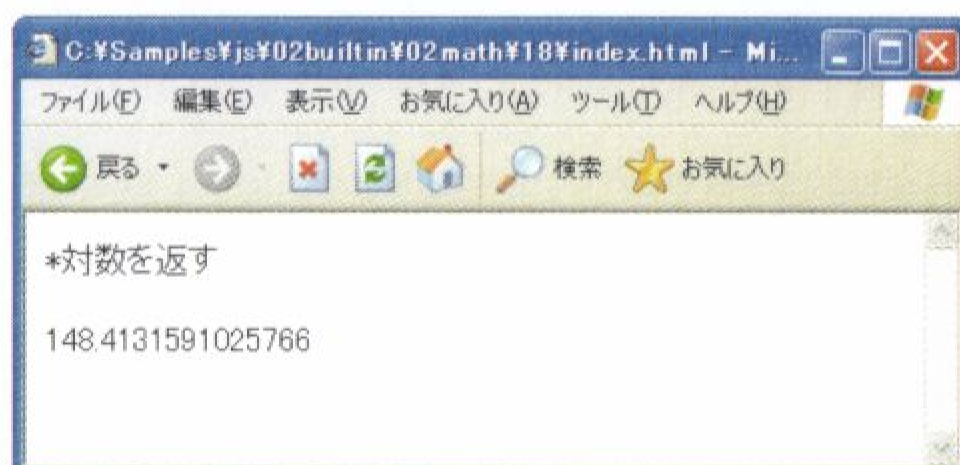
**Sample**

```
<script type="text/javascript">
<!--
document.write(Math.sqrt(2));
//-->
</script>
```

JavaScript

## 対数を返す

**Math.exp(*n*)** [メソッド]



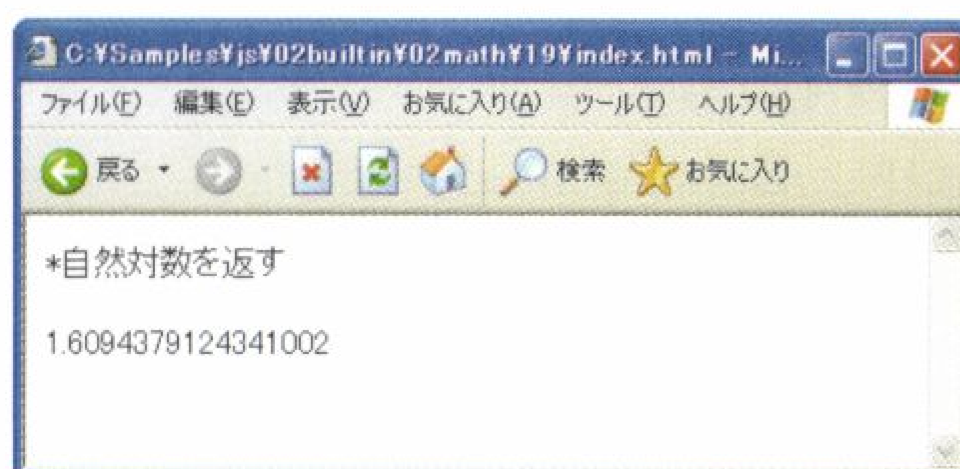

「exp()」メソッドは、nの対数を返します。

**Sample**

```
<script type="text/javascript">
<!--
document.write(Math.exp(5));
//-->
</script>
```

# 自然対数を返す

**Math.log(*n*)** ［メソッド］




「log()」メソッドは、nの自然対数を返します。

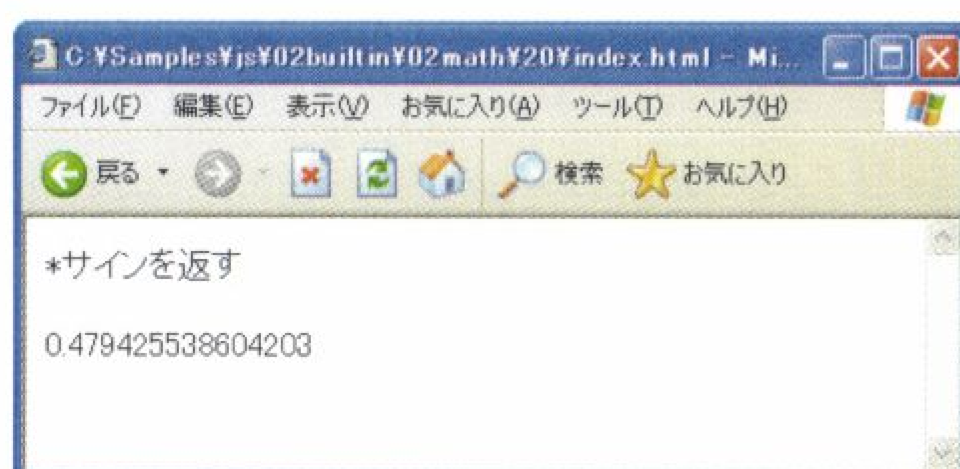
**Sample**

```
<script type="text/javascript">
<!--
document.write(Math.log(5));
//-->
</script>
```



# サインを返す

**Math.sin(*n*)** ［メソッド］




「sin()」メソッドは、nのサイン(正弦)の値を返します。単位はラジアンです。

**Sample**

```
<script type="text/javascript">
<!--
document.write(Math.sin(0.5));
//-->
</script>
```

JavaScript

# コサインを返す

**Math.cos(*n*)** [メソッド]




「cos()」メソッドは、nのコサイン(余弦)の値を返します。単位はラジアンです。

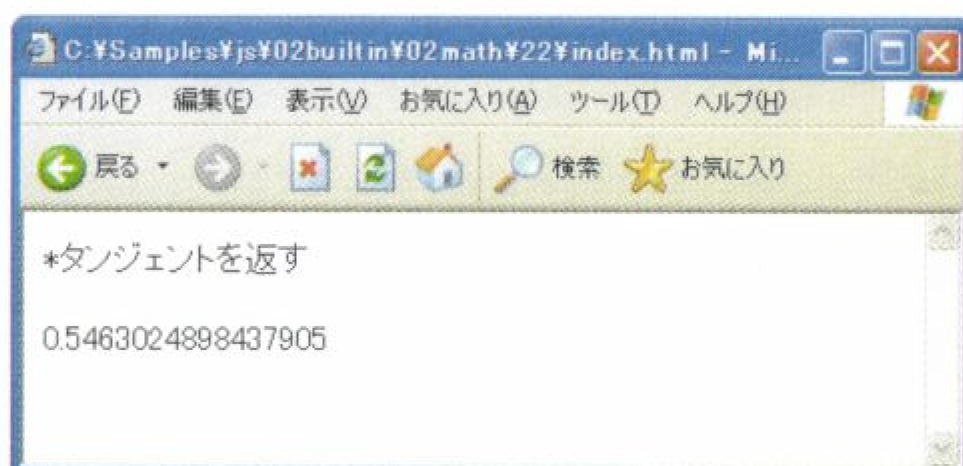
**Sample**

```
<script type="text/javascript">
<!--
document.write(Math.cos(0.5));
//-->
</script>
```

JavaScript

# タンジェントを返す

**Math.tan(*n*)** [メソッド]




「tan()」メソッドは、nのタンジェント(正接)の値を返します。単位はラジアンです。

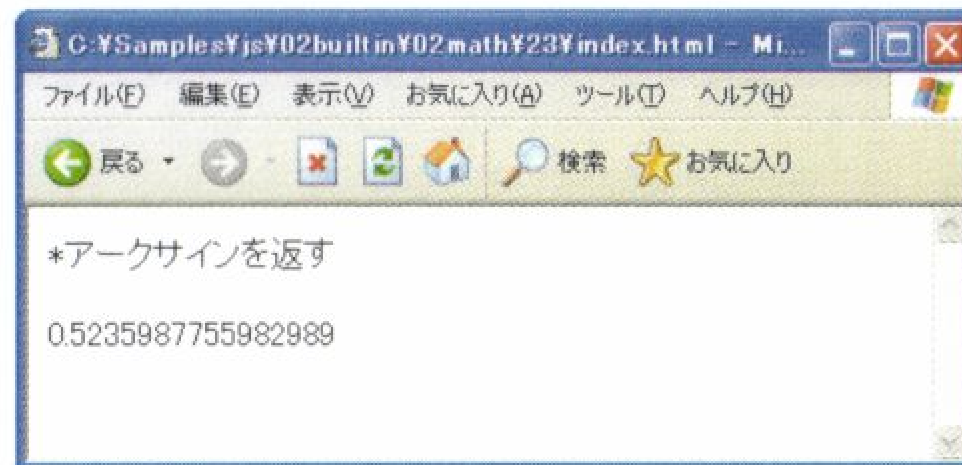
**Sample**

```
<script type="text/javascript">
<!--
document.write(Math.tan(0.5));
//-->
</script>
```

# アークサインを返す

**Math.asin(*n*)** [メソッド]






「asin()」メソッドは、nのアークサイン(逆正弦)の値を返します。単位はラジアンです。nには-1から1までの数値が入り、それ以外の場合には0を返します。

## Sample

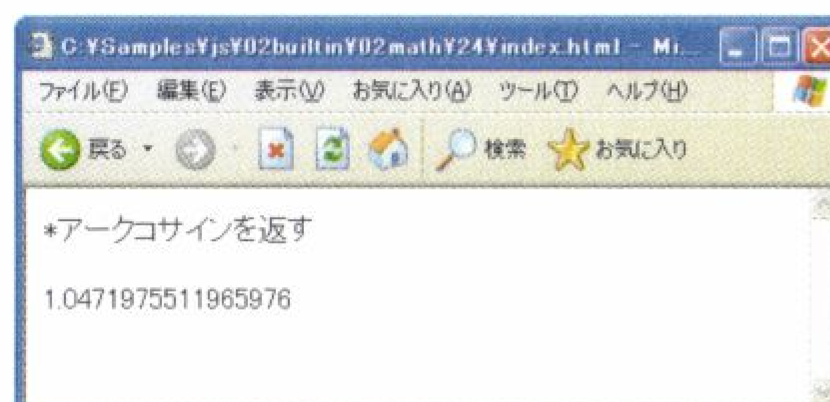
```
<script type="text/javascript">
<!--
document.write(Math.asin(0.5));
//-->
</script>
```



JavaScript

# アークコサインを返す

**Math.acos(*n*)** [メソッド]



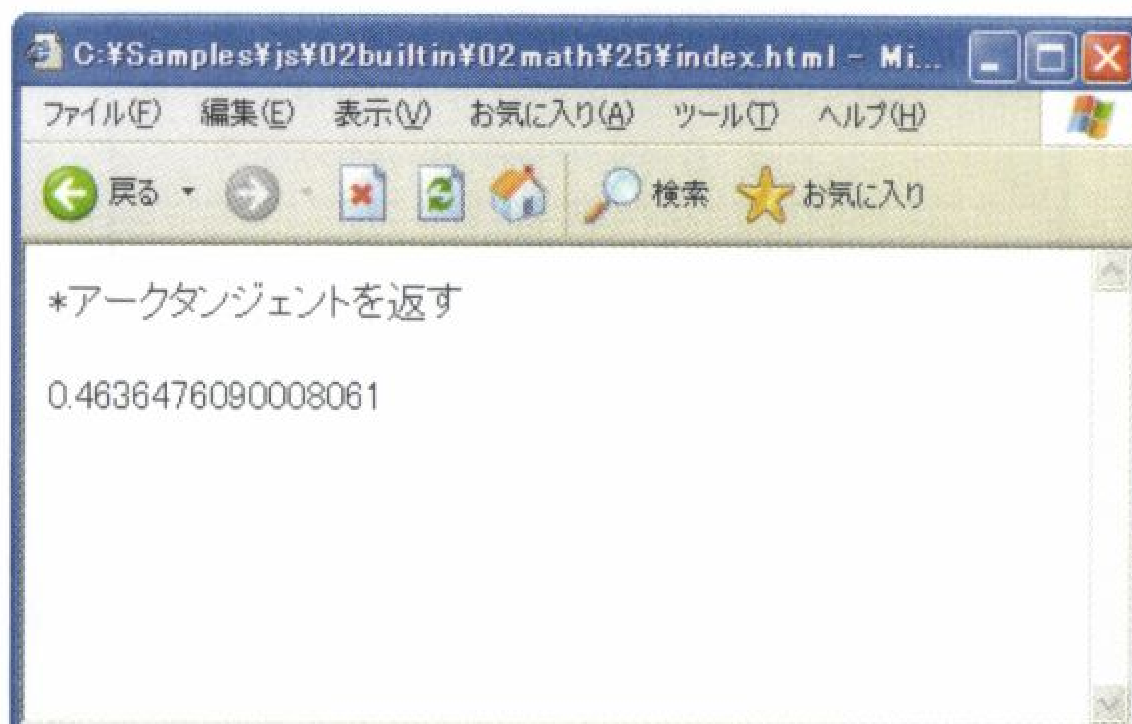

「acos()」メソッドは、nのアークコサイン(逆余弦)の値を返します。単位はラジアンです。nには-1から1までの数値が入り、それ以外の場合には0を返します。

## Sample

```
<script type="text/javascript">
<!--
document.write(Math.acos(0.5));
//-->
</script>
```

# アークタンジェントを返す

**Math.atan(*n*)** [メソッド]




「atan()」メソッドは、nのアークタンジェント（逆正接）の値を返します。単位はラジアンです。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*アークタンジェントを返す
<p>

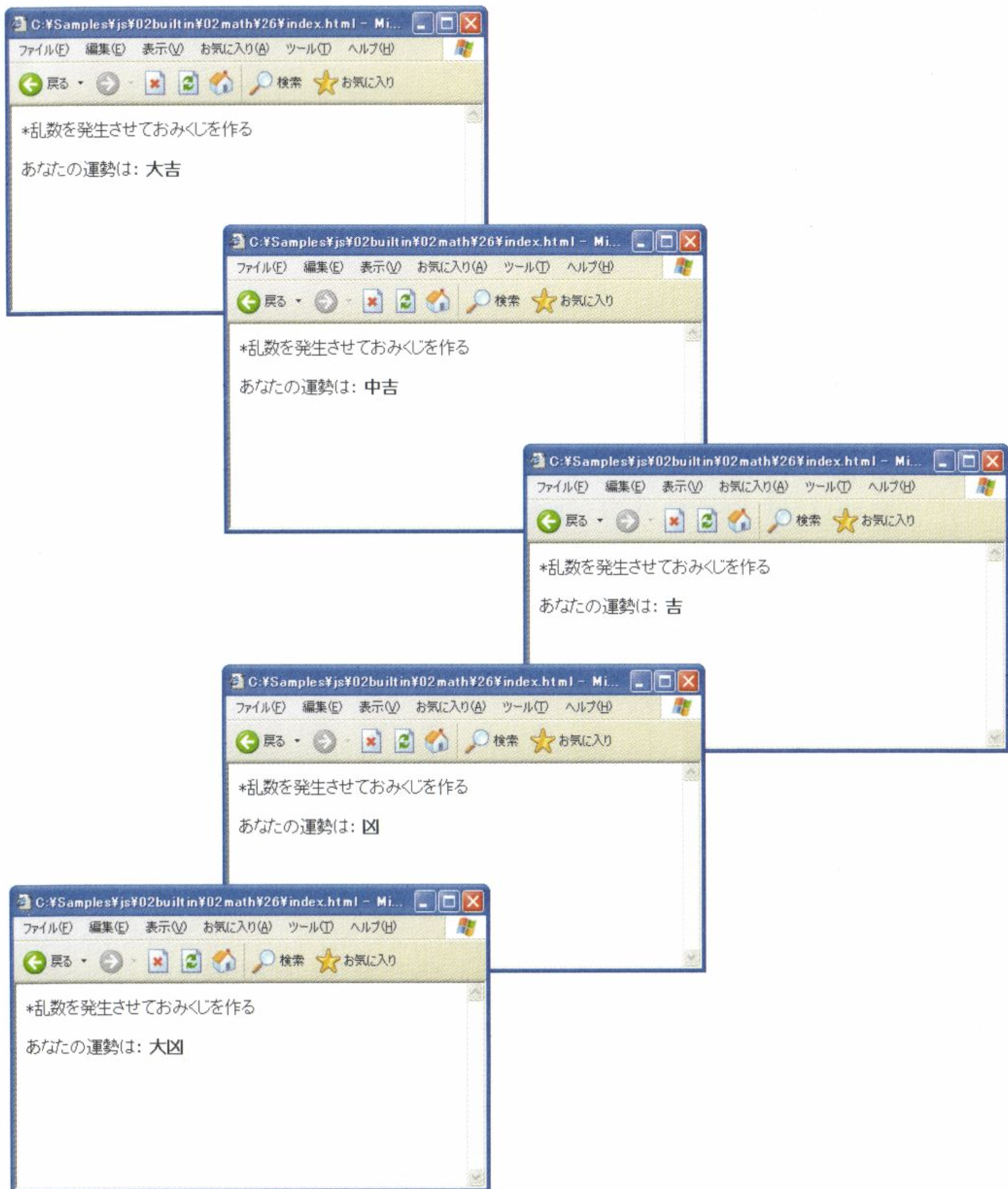
<script type="text/javascript">
<!--
document.write(Math.atan(0.5));
//-->
</script>

</p>
</body>
</html>
```

# 乱数を発生させておみくじを作る

## Math.random()　[メソッド]






「random()」メソッドは、0から1までの乱数を返します。
サンプルでは、「Math.random()」で発生した乱数の値を、関数「Unsei()」内の処理で参照し、その値によって書き出す文字を変更することによって、ページがロードされるごとにランダムに表示される文字が変わるおみくじを作っています。
Netscape Navigator 2.Xは、UNIX版のみ作動します。Windows版のInternet Explorer 3.0以降と、すべてのOSのNetscape Navigator 3.0以降のブラウザで、正常に作動します。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
var Omikuzi0 = "大吉";
var Omikuzi1 = "中吉";
var Omikuzi2 = "小吉";
var Omikuzi3 = "吉";
var Omikuzi4 = "末吉";
var Omikuzi5 = "凶";
var Omikuzi6 = "大凶";
function Unsei(n){
    if (n<=0.15) document.write( Omikuzi0.bold() );
        else { if (n<=0.3) document.write( Omikuzi1.bold() );
        else { if (n<=0.45) document.write( Omikuzi2.bold() );
        else { if (n<=0.6) document.write( Omikuzi3.bold() );
        else { if (n<=0.75) document.write( Omikuzi4.bold() );
        else { if (n<=0.9) document.write( Omikuzi5.bold() );
        else { if (n<=1) document.write( Omikuzi6.bold() );
    }}}}}}
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*乱数を発生させておみくじを作る
<p>
あなたの運勢は:

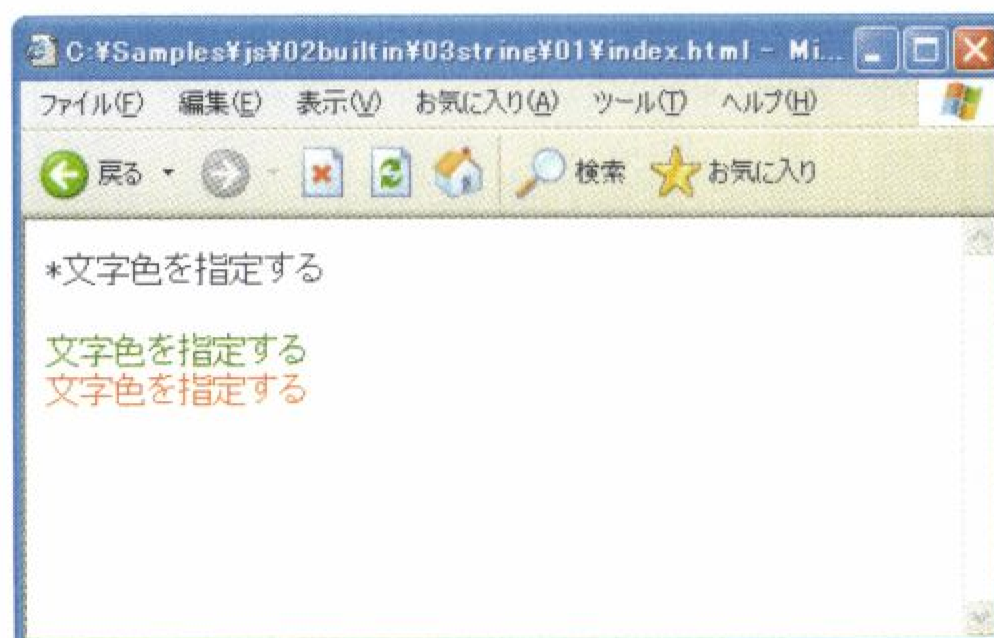
<script type="text/javascript">
<!--
Unsei(Math.random())
//-->
</script>

</p>
</body></html>
```

# 文字色を指定する

**文字列.fontcolor("色指定")** [メソッド]






「fontcolor()」メソッドは、文字列の色を指定します。
<font color="色指定">と同じ働きをします。
色指定は、色の名前か16進数で設定します。

## Sample

```
<script type="text/javascript">
<!--
document.write("文字色を指定する".fontcolor("green"));
document.write("<br>");
document.write("文字色を指定する".fontcolor("FF0000"));
//-->
</script>
```

色指定：「巻末付録」の「カラーチャート１～3」(巻末)

## TIPS

### 文字列に複数の効果を与えたい時は

文字列を修飾する時に複数の効果を同時に出したい場合は、文字列の後にその効果を与えるメソッドを並べて記述します。

たとえば、文字列を太文字にして、色を付け、斜体文字にする場合は、<script>内で次のように記述します。

```
document.write("文字列".bold().fontcolor("green").italics())
```

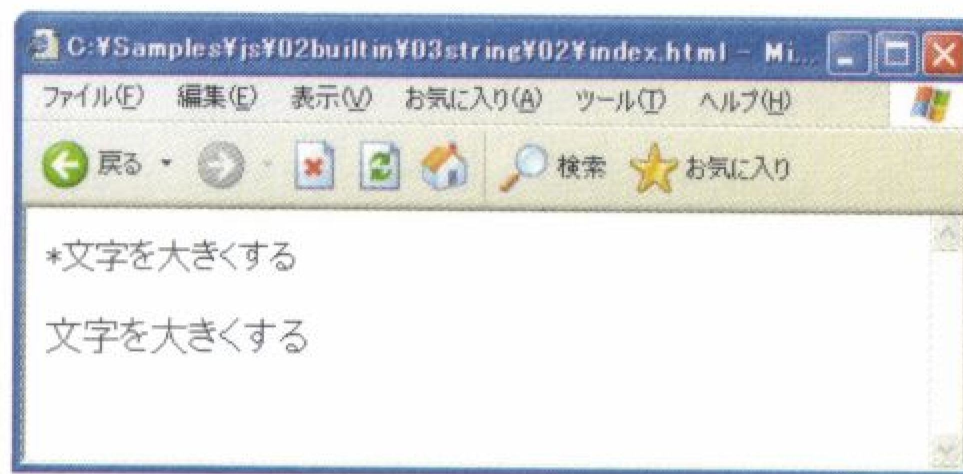
ちなみに、このスクリプトは、HTMLで次のように記述するのと同じです。

```
<b><font color="green"><i>文字列</i></font></b>
```

JavaScript

# 文字を大きくする

**文字列.big()** [メソッド]




「big()」メソッドは、文字列を通常より大きな文字で表示します。
<big>と同じ働きをします。

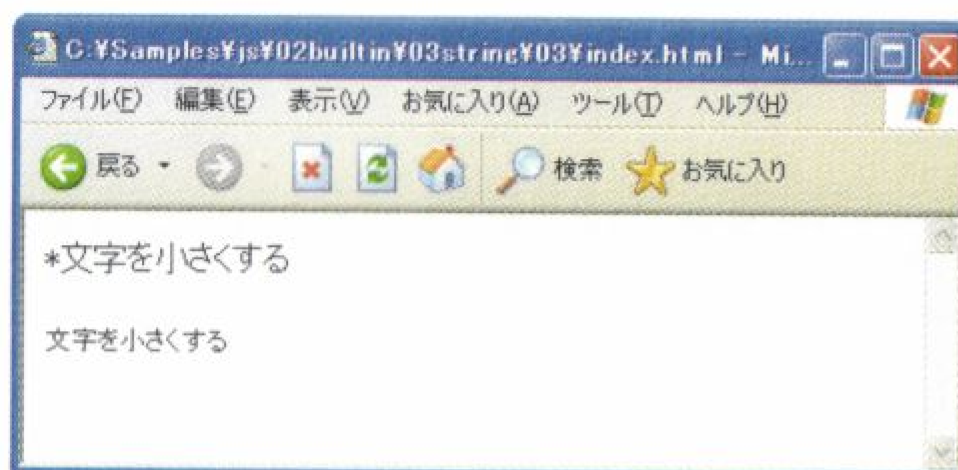
**Sample**

```
<script type="text/javascript">
<!--
document.write("文字を大きくする".big());
//-->
</script>
```

JavaScript

# 文字を小さくする

**文字列.small()** [メソッド]




「small()」メソッドは、文字列を通常より小さな文字で表示します。
<small>と同じ働きをします。

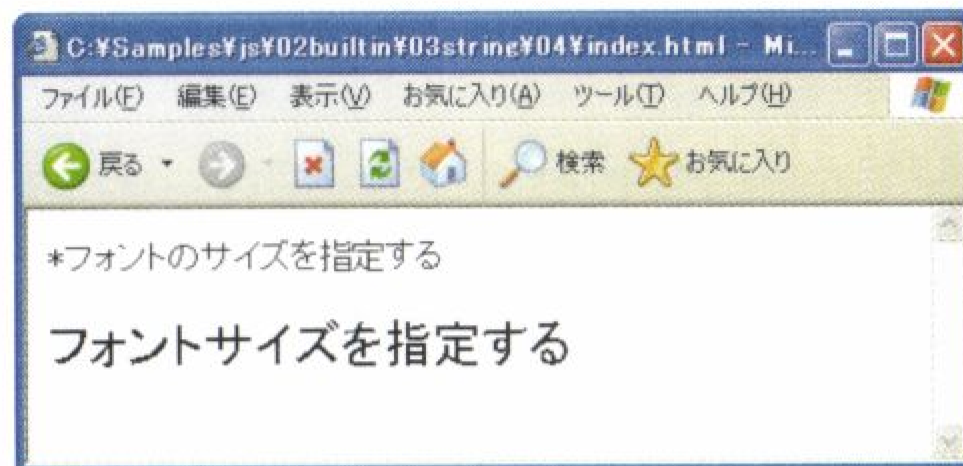
**Sample**

```
<script type="text/javascript">
<!--
document.write("文字を小さくする".small());
//-->
</script>
```

JavaScript

# フォントのサイズを指定する

**文字列.fontsize(n)** [メソッド]




「fontsize()」メソッドは、文字列のフォントサイズを指定します。
<font size=n>と同じ働きをします。

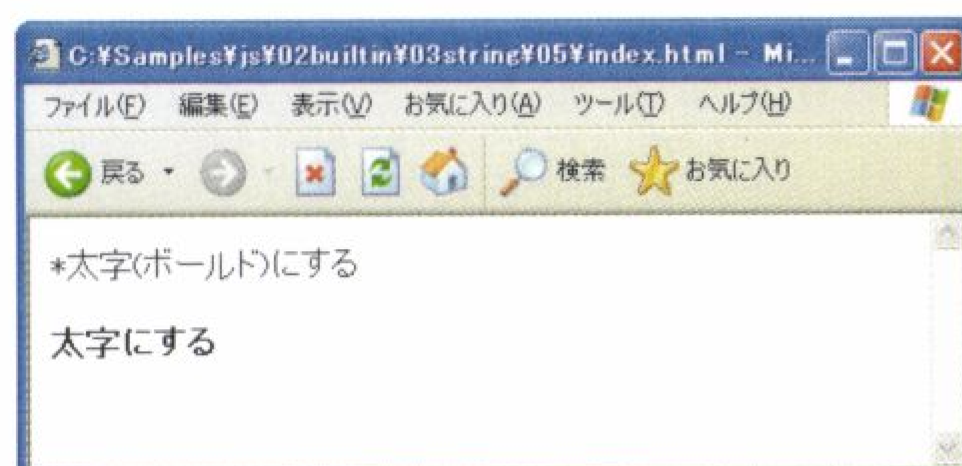
**Sample**

```
<script type="text/javascript">
<!--
document.write("フォントサイズを指定する".fontsize(5));
//-->
</script>
```

JavaScript

# 太字（ボールド）にする

**文字列.bold()** [メソッド]




「bold()」メソッドは、文字列を太文字(ボールド)にします。
<b>と同じ働きをします。

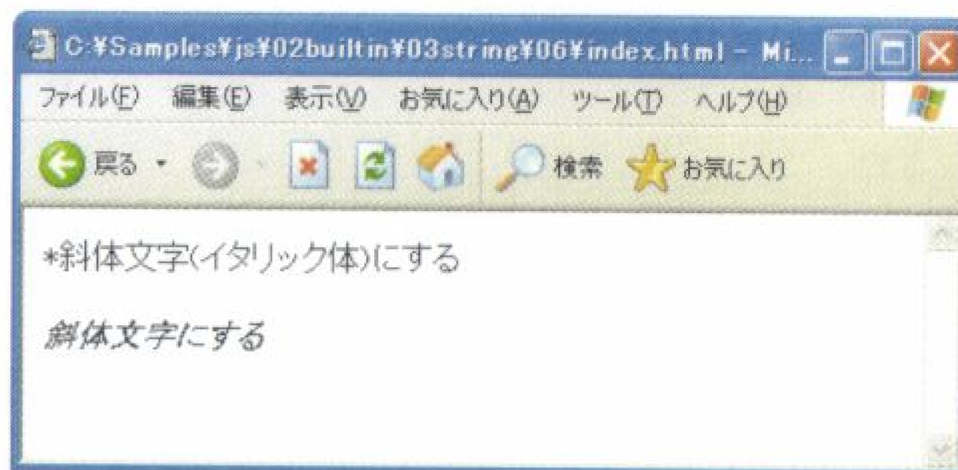
**Sample**

```
<script type="text/javascript">
<!--
document.write("太字にする".bold());
//-->
</script>
```

JavaScript

## 斜体文字(イタリック)にする

文字列.italics()　[メソッド]




「italics()」メソッドは、文字列を斜体文字にします。
<i>と同じ働きをします。

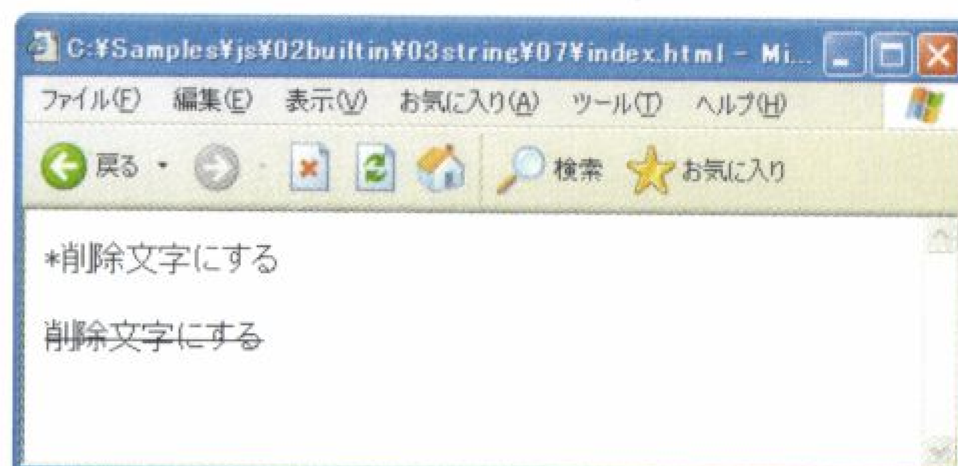
**Sample**

```
<script type="text/javascript">
<!--
document.write("斜体文字にする".italics());
//-->
</script>
```

JavaScript

## 削除文字にする

文字列.strike()　[メソッド]




「strike()」メソッドは、文字列を削除文字にします。
<strike>と同じ働きをします。

**Sample**

```
<script type="text/javascript">
<!--
document.write("削除文字にする".strike());
//-->
</script>
```

JavaScript

# 上付文字にする

文字列.sup() [メソッド]




「sup()」メソッドは、文字列を上付文字にします。
<sup>と同じ働きをします。

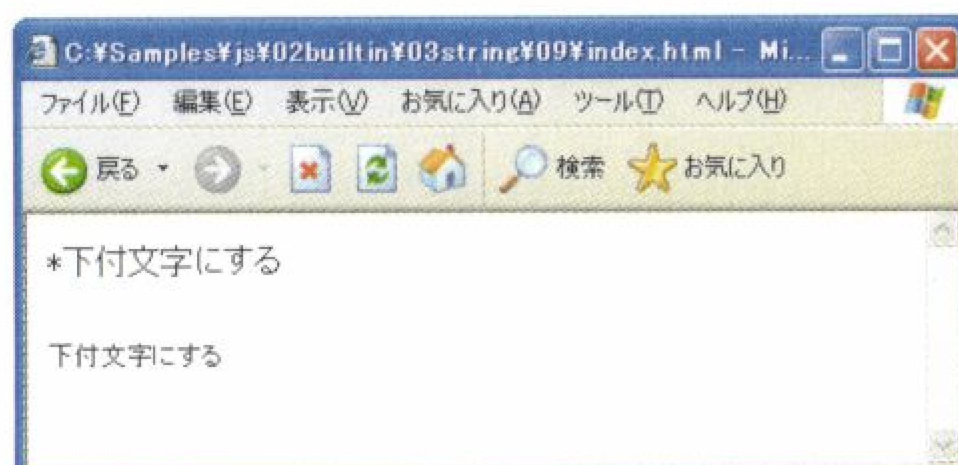
**Sample**

```
<script type="text/javascript">
<!--
document.write("上付文字にする".sup());
//-->
</script>
```

JavaScript

# 下付文字にする

文字列.sub() [メソッド]




「sub()」メソッドは、文字列を下付文字にします。
<sub>と同じ働きをします。

**Sample**

```
<script type="text/javascript">
<!--
document.write("下付文字にする".sub());
//-->
</script>
```

JavaScript

# 等幅文字にする

## 文字列.fixed() [メソッド]




「fixed()」メソッドは、文字列を等幅フォントで表示します。
<tt>と同じ働きをします。

**Sample**

```
<script type="text/javascript">
<!--
document.write("等幅文字にする".fixed());
//-->
</script>
```

JavaScript

# 文字を点滅する

## 文字列.blink() [メソッド]







「blink()」メソッドは、文字列を点滅させます。
<blink>と同じ働きをします。Netscape7.X、Firefoxでは対応していますが、Internet Explorer や Netscape 6.X、Safari は、<blink>に未対応なため何も反応しません。

**Sample**

```
<script type="text/javascript">
<!--
document.write("文字を点滅する".blink());
//-->
</script>
```

# リンクを作る

**文字列.link(*URL*)** [メソッド]



Before




After




「link()」メソッドは、リンクを設定します。
<a href ="URL"> と同じ働きをします。

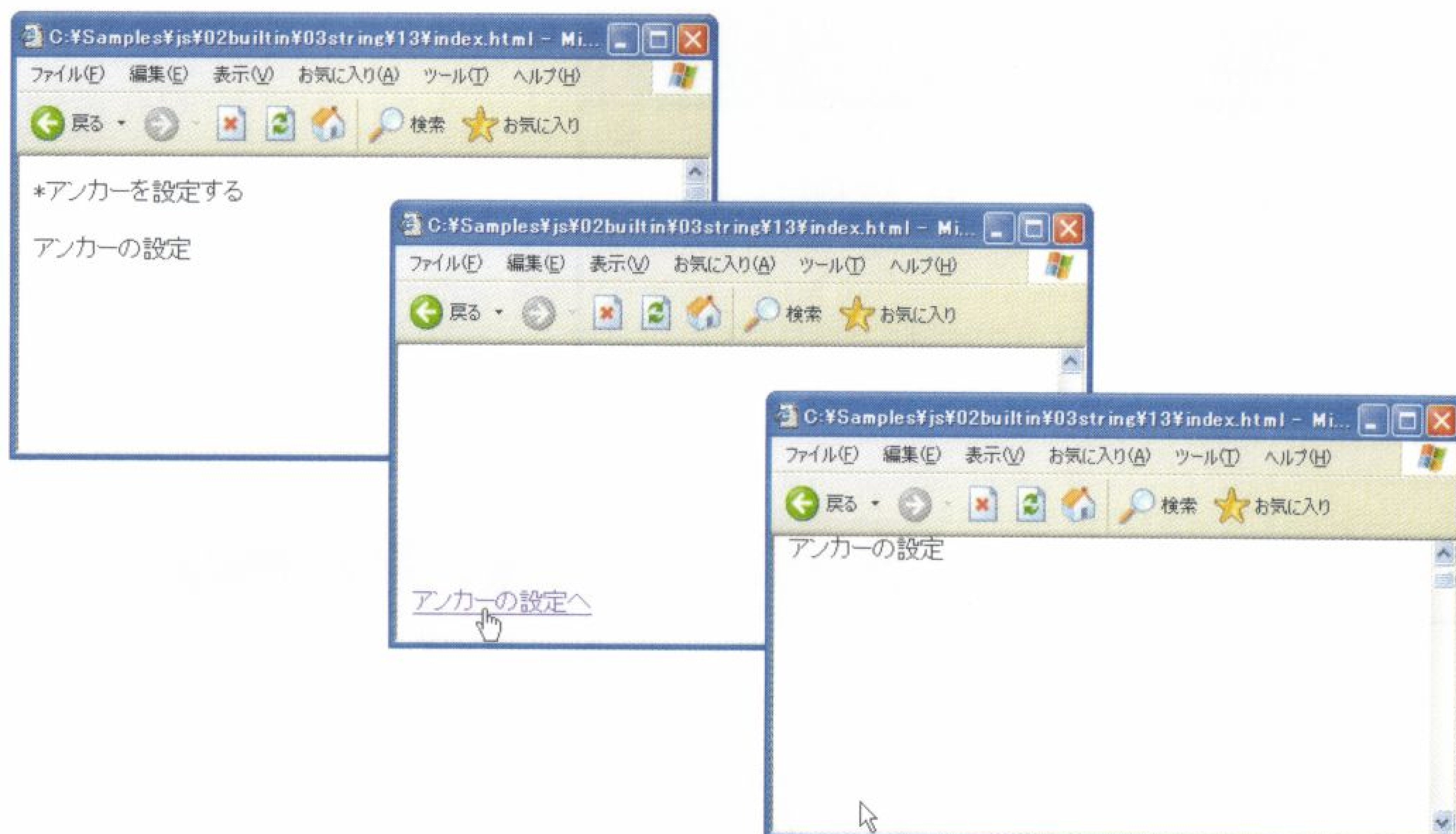
## Sample

```
<script type="text/javascript">
<!--
document.write("リンクを作る".link("http://www.shuwasystem.co.jp/"));
//-->
</script>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# アンカーを設定する

## 文字列.anchor() [メソッド]




「anchor()」メソッドは、アンカーの設定をします。
<a name="">と同じ働きをします。

### Sample

```
<body>
<p>*アンカーを設定する</p>

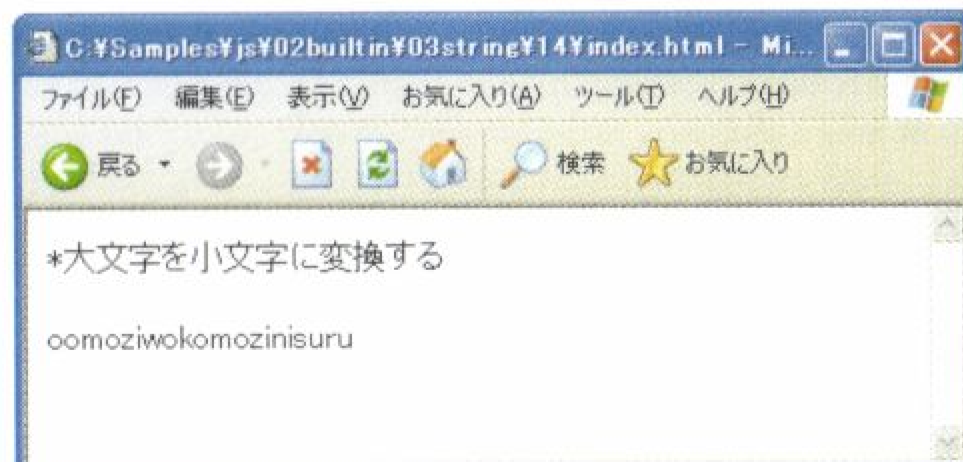
<script type="text/javascript">
<!--
document.write("アンカーの設定".anchor("test1"));
//-->
</script>

<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
r><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
r><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
r><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
r><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br>
<a href="#test1">アンカーの設定へ</a>
</body>
```

JavaScript

# 大文字を小文字に変換する

文字列.toLowerCase()　[メソッド]




「toLowerCase()」メソッドは、大文字の文字列を小文字に変換します。

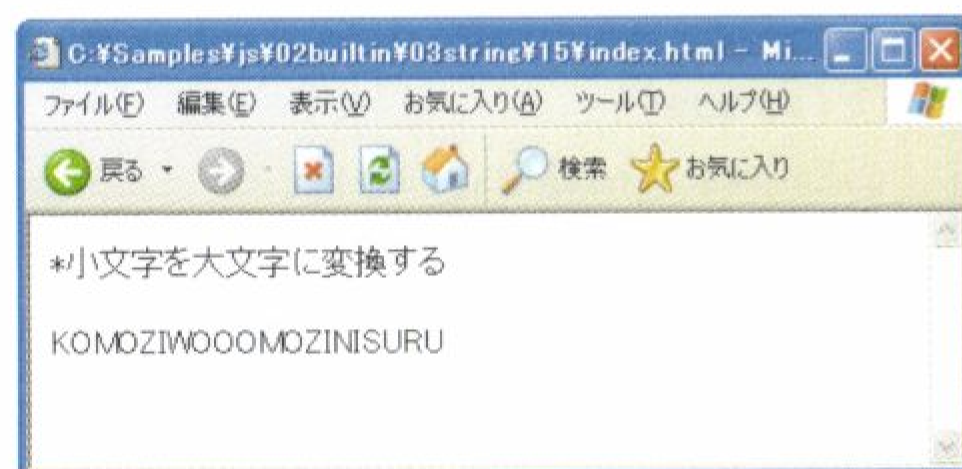
## Sample

```
<script type="text/javascript">
<!--
document.write("OOMOZIWOKOMOZINISURU".toLowerCase());
//-->
</script>
```

JavaScript

# 小文字を大文字に変換する

文字列.toUpperCase()　[メソッド]



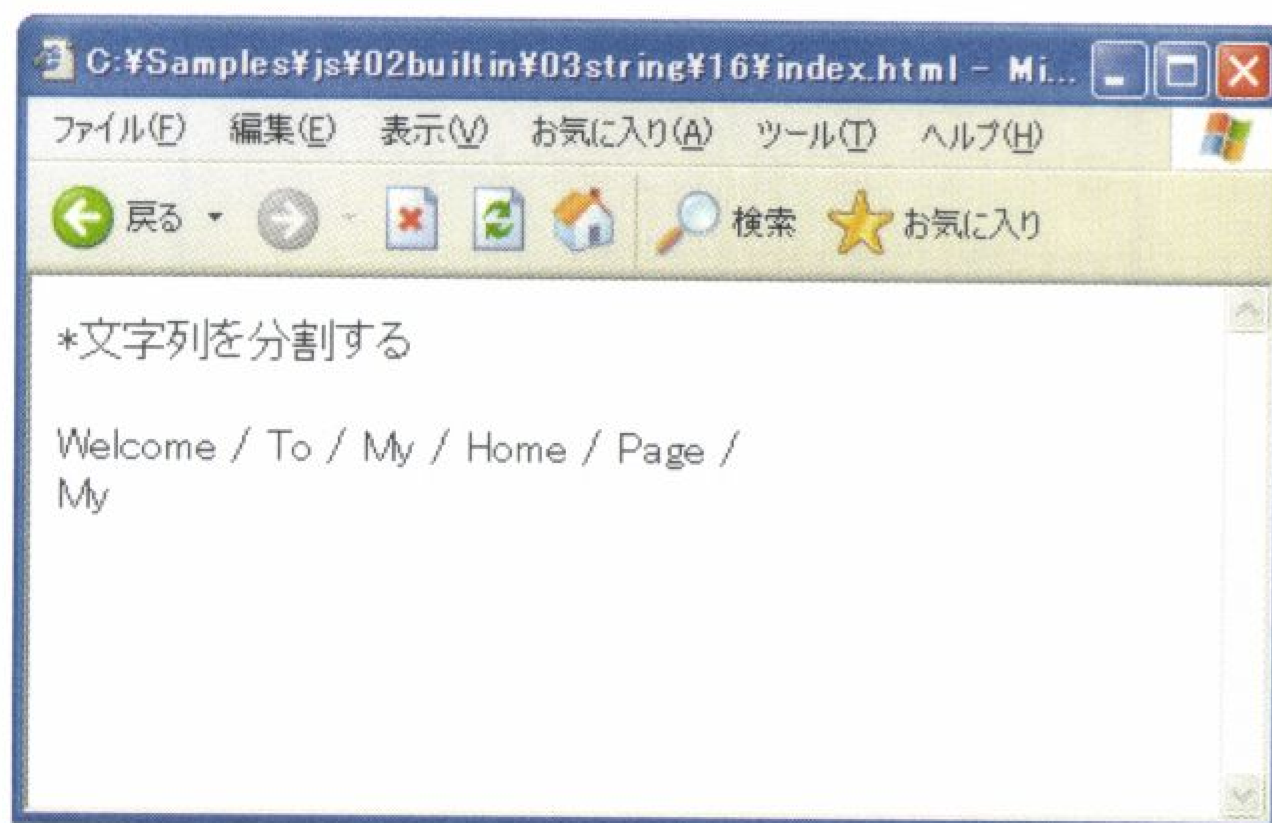

「toUpperCase()」メソッドは、小文字の文字列を大文字に変換します。

## Sample

```
<script type="text/javascript">
<!--
document.write("komoziwooomozinisuru".toUpperCase());
//-->
</script>
```

# 文字列を分割する

文字列.split(分割する文字列) [メソッド]




「split()」メソッドは、文字列を指定した文字の部分で分割します。
サンプルでは、「Welcome To My Home Page」の各スペースごとに文字列を分割し、「/」を挿入しています。
また、分割された文字列は0から始まる配列になります。したがって、「SPmozi[2]」は「My」を返します。
JavaScript1.1で追加されたメソッドです。

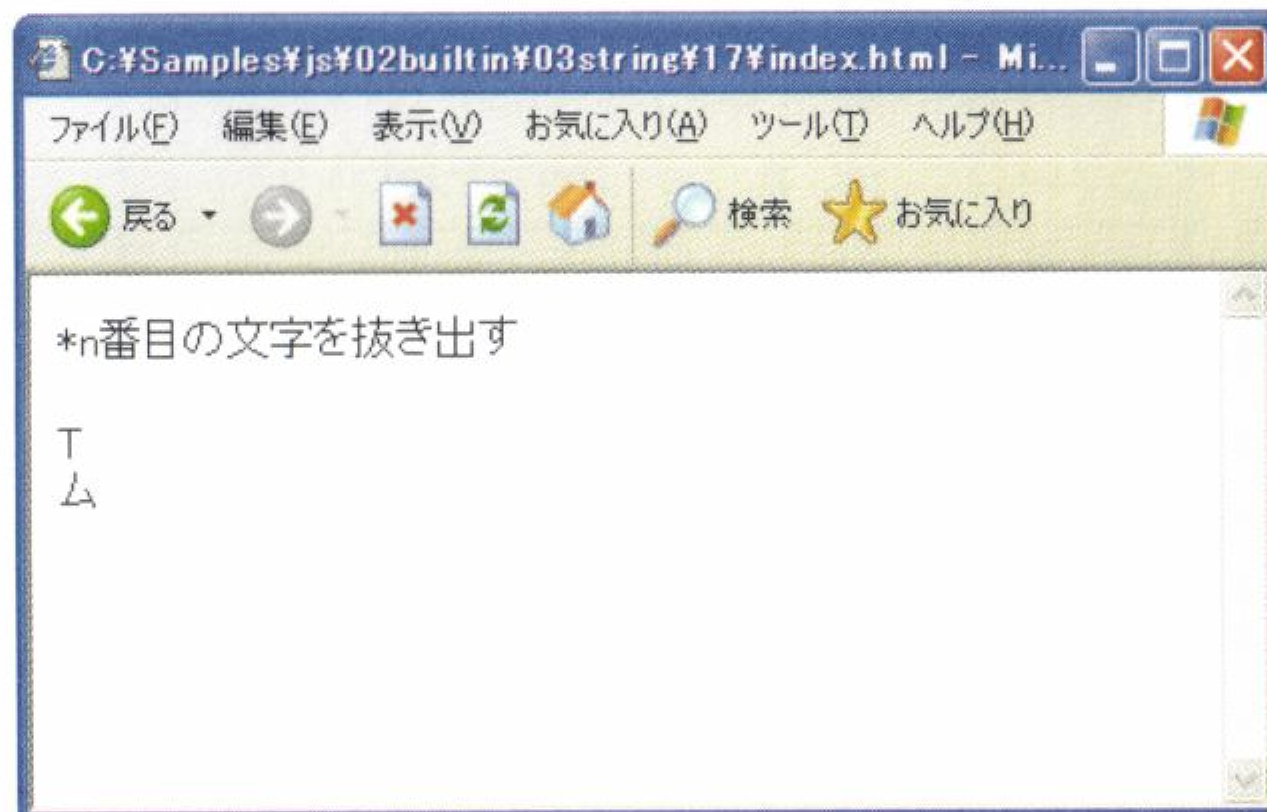
## Sample

```
<script type="text/javascript">
<!--
var mozi="Welcome To My Home Page";
var space=" ";
var SPmozi = mozi.split(space);
    for (var i=0; i < SPmozi.length; i++) {
        document.write (SPmozi[i] + " / ");
    }
document.write ("<br>");
document.write (SPmozi[2]);
//-->
</script>
```

# n番目の文字を抜き出す

文字列.charAt(n) [メソッド]





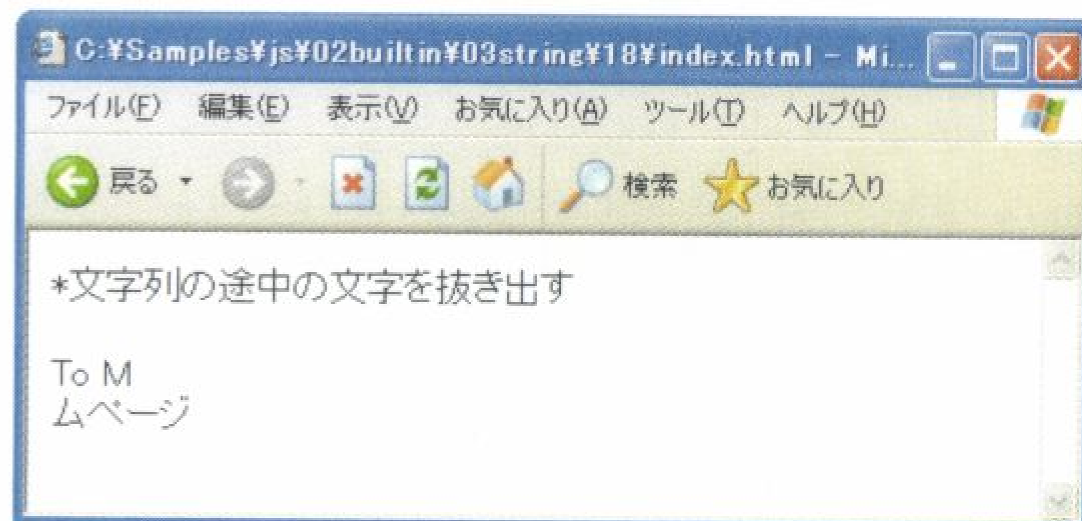

「charAt()」メソッドは、0スタートの先頭から数えてn番目の文字を抜き出します。
サンプルでは、「Welcome To My Home Page」の9番目の文字である「T」が抜き出されます。
Netscape Navigatorの場合は、JavaScript1.2までは日本語のような2バイト文字は1文字を2字として見るため、日本語の文字列を正確に抜き出すことができませんでした。しかし、JavaScript1.3からは文字コードがUnicodeになったので、日本語も正確に抜き出されます。
なお、Internet Explorerは、日本語も半角英数字も1文字としてカウントします。

## Sample

```
<script type="text/javascript">
<!--
var mozi1 = "Welcome To My Home Page";
var mozi2 = "ようこそ私のホームページへ";
    document.write(mozi1.charAt(8));
    document.write("<br>");
    document.write(mozi2.charAt(8));
//-->
</script>
```

# 文字列の途中の文字を抜き出す

**文字列.substring(n,m)** [メソッド]




「substring()」メソッドは、0スタートの先頭から数えて、n番目の文字からm番目の文字の直前までの文字列を抜き出します。

Internet Explorerでは、日本語も半角英数字も1文字としてカウントします。Netscape Navigatorの場合は、JavaScript1.2までは日本語のような2バイト文字は1文字を2字としてカウントしていましたが、JavaScript1.3からは文字コードがUnicodeになったので、日本語も1文字を1字としてカウントするようになっています。

## Sample

```
<script type="text/javascript">
<!--
var mozi1 = "Welcome To My Home Page";
var mozi2 = "ようこそ私のホームページへ";
    document.write(mozi1.substring(8,12));
    document.write("<br>");
    document.write(mozi2.substring(8,12));
//-->
</script>
```

### 重要

#### 日本語(2バイト文字)の取り扱いについて

Netscape Navigatorでは、JavaScript1.2までは文字を常に1バイト文字として認識していたので、日本語のひらがなや漢字などの2バイト文字は1文字を2字と認識していました。それに対して日本語版のInternet Explorerは、1バイト文字も2バイト文字も1文字は1字として認識します。

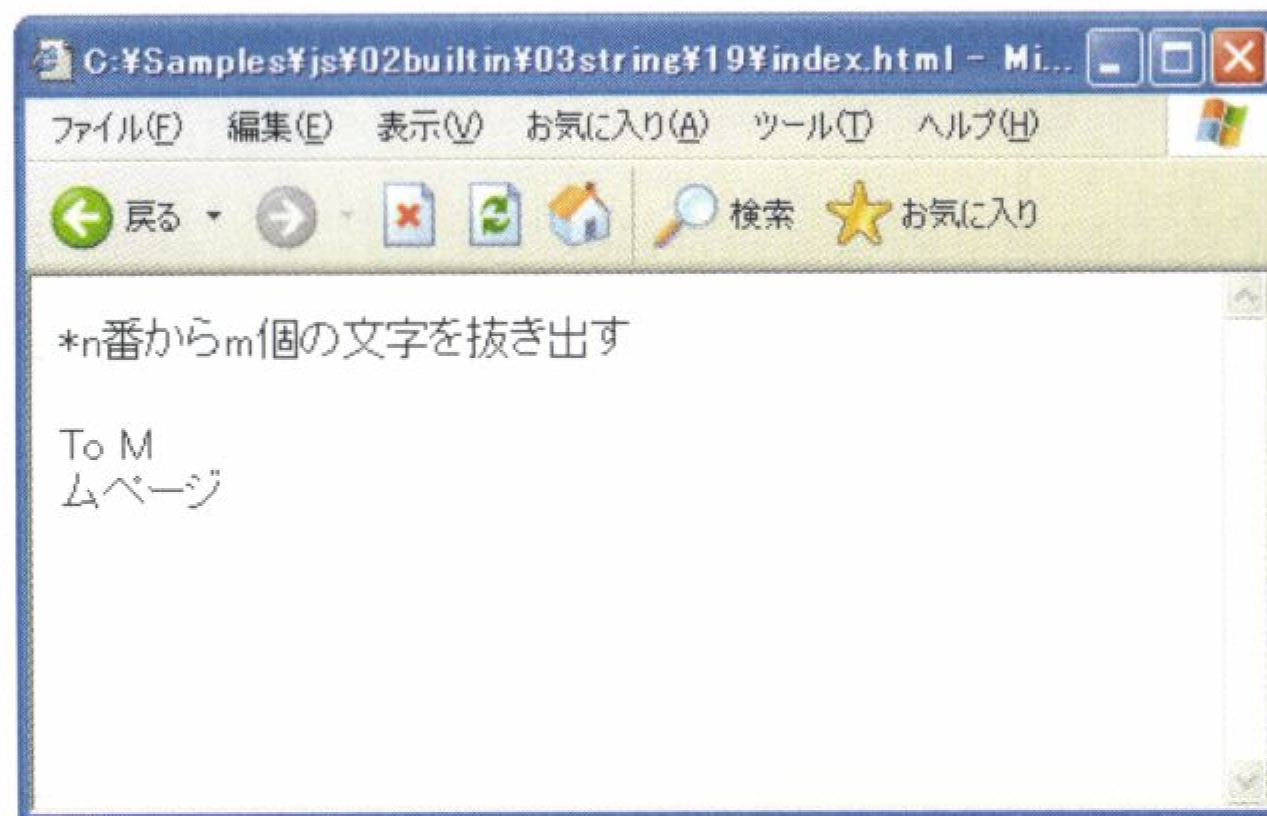
このため、日本語に対して「indexOf()」メソッドや「substring()」メソッドなどの文字を検索したり抜き出すスクリプトを使用したり、「length」プロパティで文字の数を調べたりすると、Netscape NavigatorとInternet Explorer(日本語版)では異なった結果になりました。

JavaScript1.3からは、文字コードがUNICODEになったので、Netscape Navigatorでも日本語も1文字を1字としてカウントするようになりました。

# n番からm個の文字を抜き出す

**文字列.substr(n,m)** [メソッド]






「substr()」メソッドは、0スタートの先頭から数えて、n番目の文字からm個の文字を抜き出します。n番目からm番目の文字を抜き出す、JavaScript1.0の「substring()」メソッド(前項の「文字列の途中の文字を抜き出す」)との違いに注意してください。
Internet Explorerでは、日本語も半角英数字も1文字としてカウントします。Netscape Navigatorの場合は、JavaScript1.2までは日本語のような2バイト文字は1文字を2字としてカウントしていましたが、JavaScript1.3からは文字コードがUnicodeになったので、日本語も1文字を1字としてカウントするようになっています。
JavaScript1.2で追加されたメソッドです。

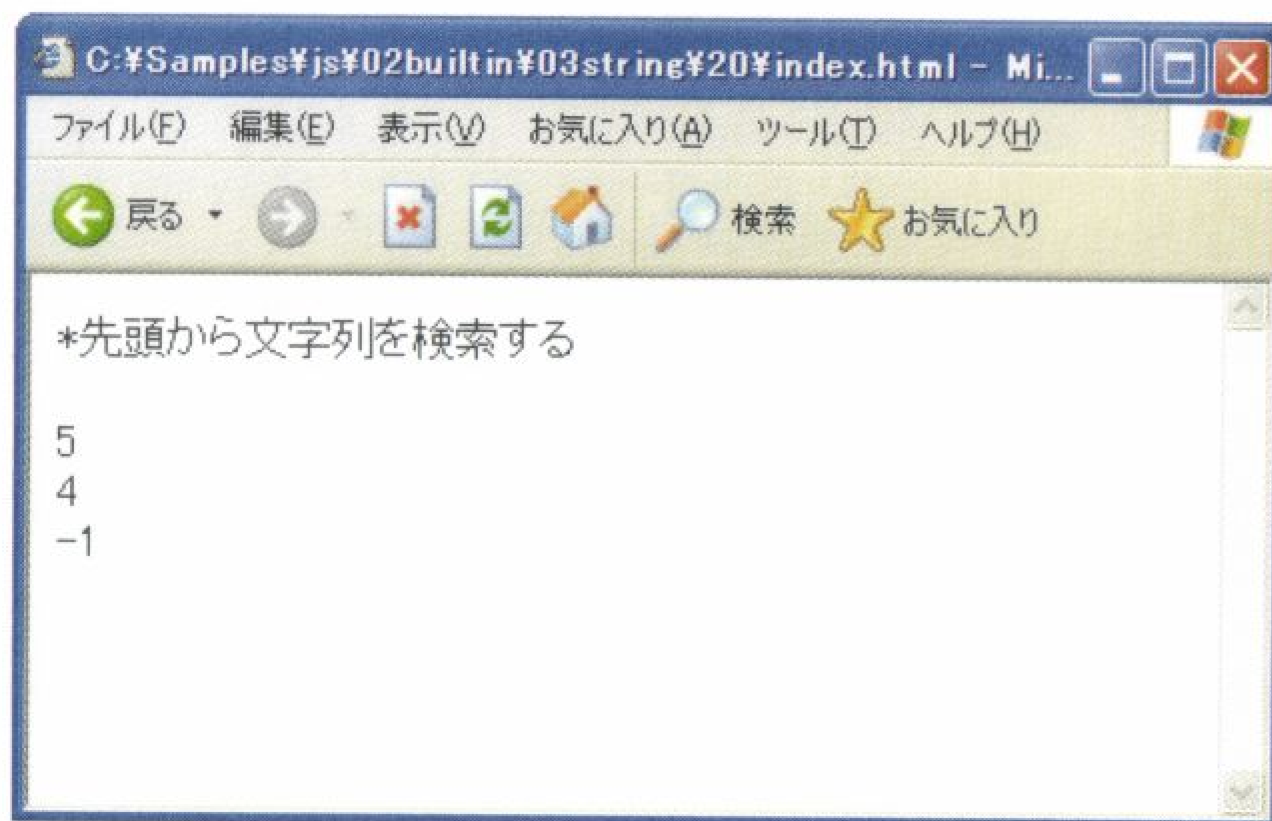
## Sample

```
<script type="text/javascript">
<!--
var mozi1 = "Welcome To My Home Page";
var mozi2 = "ようこそ私のホームページへ";
    document.write(mozi1.substr(8,4));
    document.write("<br>");
    document.write(mozi2.substr(8,4));
//-->
</script>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 先頭から文字列を検索する

文字列.indexOf(検索対象文字列,[検索開始位置]) [メソッド]




「indexOf()」メソッドは、文字列を先頭から検索します。
「検索対象文字」で指定した文字の位置を、0スタートの先頭から数えた数字で返し、もし指定した文字が含まれていない場合は「-1」を返します。
Netscape Navigatorの場合は、JavaScript1.2までは日本語のような2バイト文字は1文字を2字としてカウントしていましたが、JavaScript1.3からは文字コードがUnicodeになったので、日本語も1文字を1字としてカウントするようになっています。Internet Explorerでは、日本語も半角英数字も1文字としてカウントします。
また、「[検索開始位置]」は省略可能で、その場合は先頭から検索します。

## Sample

```
<script type="text/javascript">
<!--
var mozi1 = "Welcome To My Home Page";
var mozi2 = "ようこそ私のホームページへ";
    document.write(mozi1.indexOf("m"));
    document.write("<br>");
    document.write(mozi2.indexOf("私",0));
    document.write("<br>");
    document.write(mozi1.indexOf("A",0));
//-->
</script>
```

文字列を操作する

# 後ろから文字列を検索する

**文字列.lastIndexOf(検索対象文字列,[検索開始位置])** [メソッド]






「lastIndexOf()」メソッドは、文字列を後ろから検索します。
「検索対象文字」で指定した文字の位置を、0スタートの先頭から数えた数字で返し、もし指定した文字が含まれていない場合は「-1」を返します。
Netscape Navigatorの場合、JavaScript1.2までは日本語のような2バイト文字は1文字を2字としてカウントしていましたが、JavaScript1.3からは文字コードがUnicodeになったので、日本語も1文字を1字としてカウントするようになっています。Internet Explorerでは、日本語も半角英数字も1文字としてカウントします。
また、「[検索開始位置]」は省略可能で、その場合は1番後ろから検索します。

## Sample

```
<script type="text/javascript">
<!--
var mozi1 = "Welcome To My Home Page";
var mozi2 = "ようこそ私のホームページへ";
    document.write(mozi1.lastIndexOf("m"));
    document.write("<br>");
    document.write(mozi2.lastIndexOf("私",12));
    document.write("<br>");
    document.write(mozi1.lastIndexOf("A"));
//-->
</script>
```

# 指定した文字のISO-Latin-1のコード番号を返す

## 文字列.charCodeAt(n) [メソッド]




「charCodeAt()」メソッドは、0スタートの先頭から数えて、n番目の文字をISO-Latin-1の文字コードに変換します。
JavaScript1.2で追加されたメソッドです。

### Sample

```
<script type="text/javascript">
<!--
document.write("ABC".charCodeAt(1));
//-->
</script>
```



### TIPS

#### 文字列以外も検索・文字修飾するには

stringオブジェクトのメソッドは、"～"で括られた文字列以外でも、オブジェクトを検索したり、文字修飾の効果を与えることができます。

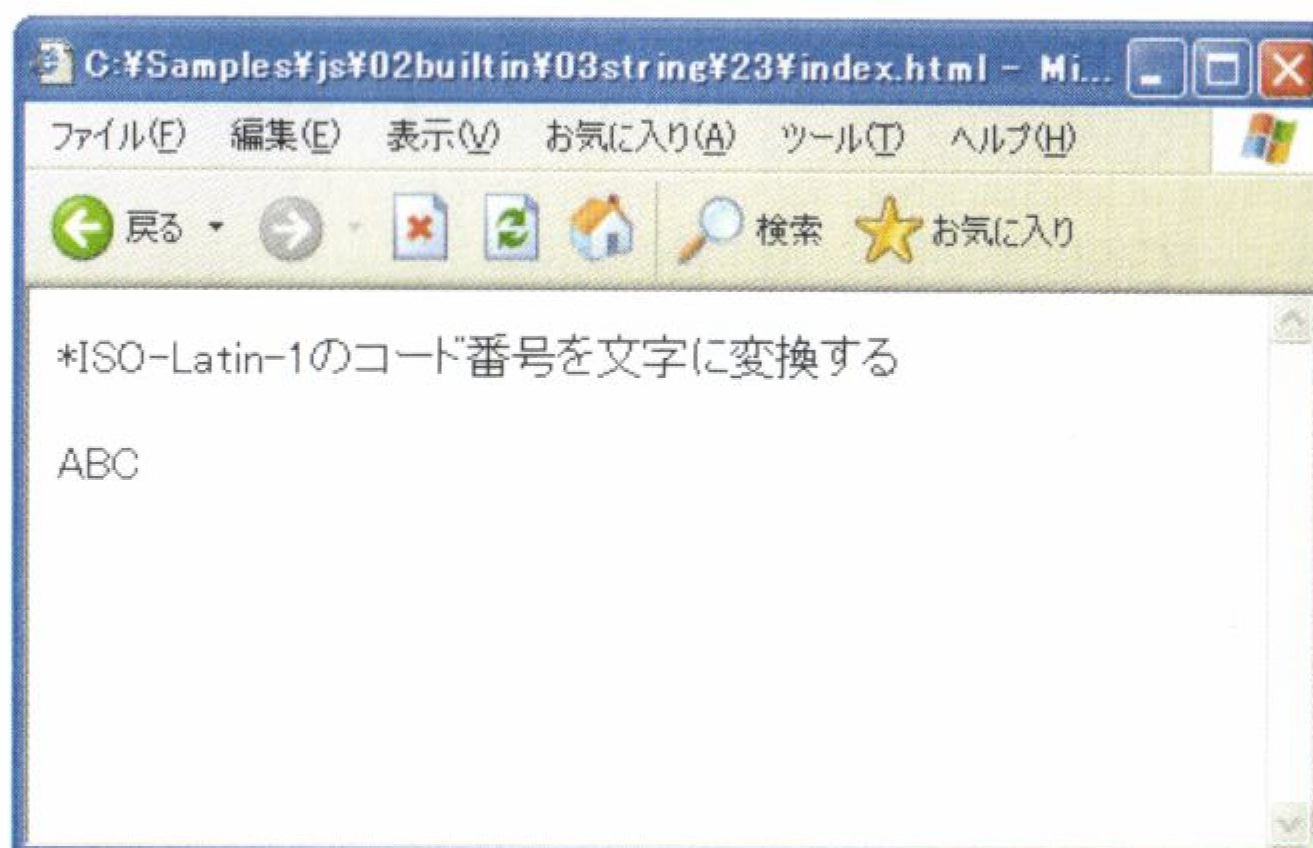
たとえばスクリプト内で、「navigator.appVersion.charAt(0)」と記述した場合は使用しているブラウザのバージョンの1番始めの数字を返し、「document.write(document.lastModified.italics())」と記述した場合はファイルの最終更新日時が斜体文字で書き出されます。

これは、各オブジェクトがstringの属性を持っているためです。

# ISO-Latin-1のコード番号を文字に変換する

**String.fromCharCode(要素1,要素2,...,要素n)** [メソッド]






「fromCharCode()」メソッドは、ISO-Latin-1のコードを文字に変換します。サンプルのように、複数のコードを「,」で区切って1度に指定できます。
JavaScript1.2で追加されたメソッドです。

## Sample

```
<script type="text/javascript">
<!--
document.write(String.fromCharCode(65,66,67));
//-->
</script>
```

## TIPS

### ISO-Latin-1コードとASCIIコード

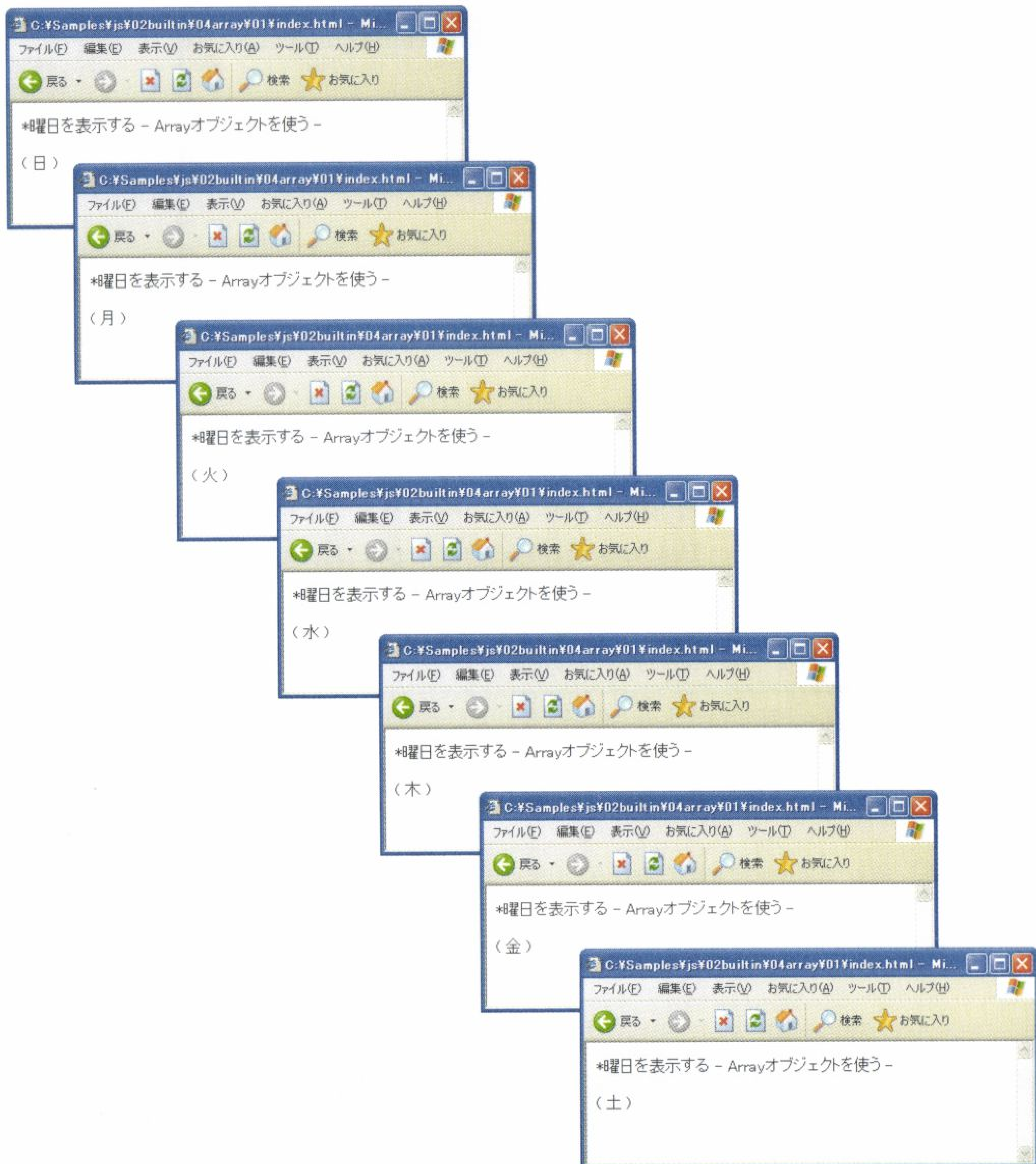
ISO-Latin-1コードは「0」から「255」まであり、そのうち「0」から「127」はASCIIのコードセットと共通しています。

したがって、「KeyDown」「KeyPress」「KeyUp」などのイベントが発生した時に、イベントが発生したキーのASCIIの値を取得できるeventオブジェクトのプロパティ「which」を使用する場合は、「String.fromCharCode(KeyDown.which)」というように「fromCharCode()」と合わせて使用すると、ASCIIコードを英数字に変換することができます。

ASCIIコードは、コンピュータ用の英数字コード体系として広く普及しているコード体系で、7bit、128文字が割り当てられています。

# 曜日を表示する - Array オブジェクトを使う -

**array オブジェクト名 = new Array(配列の数)**




サンプルでは、「Date オブジェクト」で説明した曜日を表示するスクリプト「曜日を表示する」(P.502)を、Array オブジェクトを使って書き直したものです。

「youbi= new Array(7)」で日曜日から土曜日までの7つの配列要素を持った「youbi」というオブジェクトを作成して、「getDay()」で取得した値と照合し、値が「0」の時は「youbi[0]」である「日」を、値が「1」の時は「youbi[1]」である「月」を…といった具合に配列の要素から文字列を取り出しています。

Array オブジェクトはJavaScript1.1で追加されたオブジェクトですが、このサンプルはNetscape Navigator 2.X でも使用できます。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
youbi = new Array(7);
youbi[0] = "日";
youbi[1] = "月";
youbi[2] = "火";
youbi[3] = "水";
youbi[4] = "木";
youbi[5] = "金";
youbi[6] = "土";
function gety(y){ document.write( youbi[y] ) }
//-->
</script>
 ～中略～
<body>
*曜日を表示する - Arrayオブジェクトを使う -
<p>
(
<script language="JavaScript">
<!--
day = new Date();
gety(day.getDay());
//-->
</script>
)
</p>
</body></html>
```


new Date():「Dateオブジェクト」の「年・月・日・時・分・秒を表示する」(P.500)
getDay():「Dateオブジェクト」の「曜日を表示する」(P.502)

**TIPS**

### Netscape Navigator 2.0でも使えるArrayオブジェクトは?

Arrayオブジェクトは、Netscape Navigator 3.0で対応したJavaScript1.1から追加されたオブジェクトですが、「arrayオブジェクト名 = new Array(配列の数)」の用法だけは、Netscape Navigator 2.0でも使用可能です。

しかし、それ以外の「arrayオブジェクト名 = new Array(0番目の要素,...,n番目の要素)」で作成された配列のオブジェクトや、「join()」「reverse()」「sort()」などのメソッドは使用できません。

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 配列の要素を文字列にして書き出す

**arrayオブジェクト名 = new Array(0番目の要素, 1番目の要素, ..., n番目の要素)**
**オブジェクト名.join()** [メソッド]




「join()」メソッドは、配列の要素を文字列に変換します。
「join("文字列")」と指定すると、要素の区切り時にカッコ内の文字列が加えられます。
何も指定がない時には、各要素は「,」で区切られます。
「join()」メソッドは、Netscape Navigator 2.Xでは使用できません。

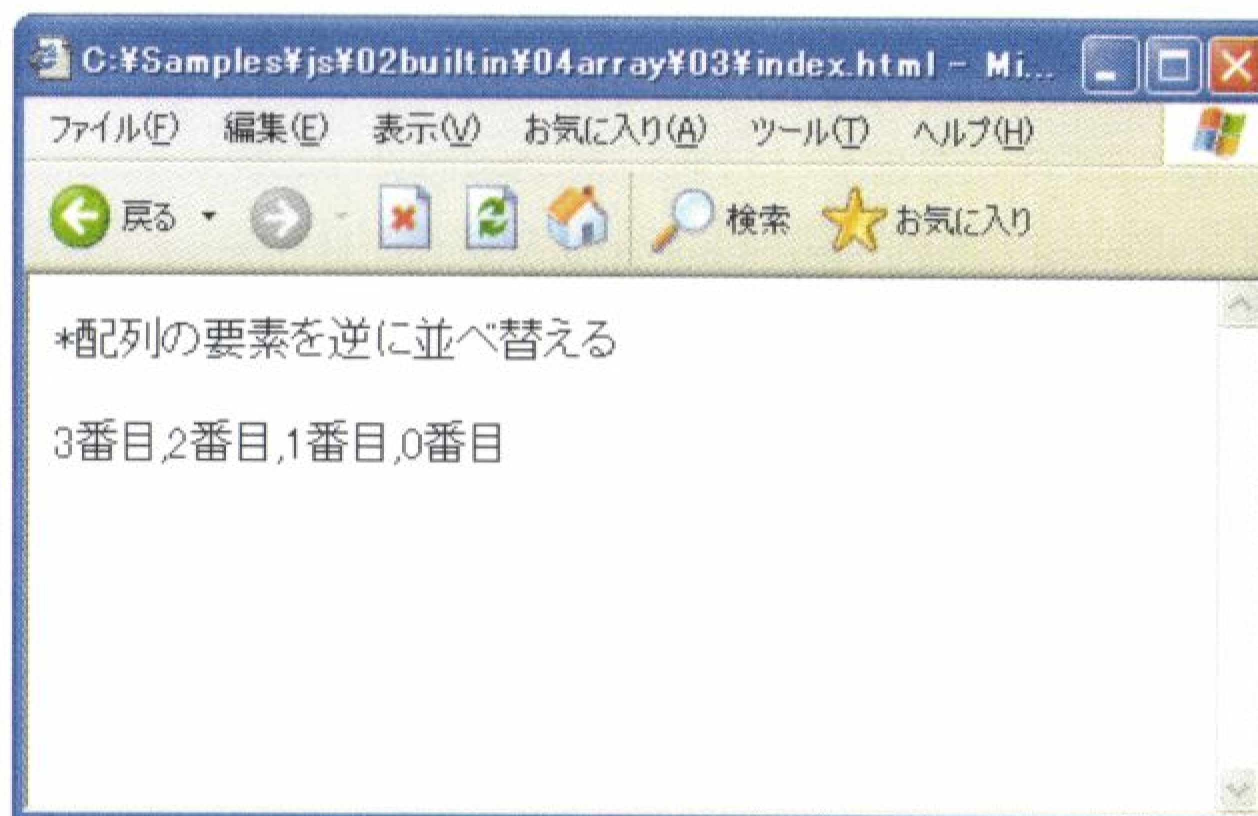
## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
  ～中略～
</head>
<body>
*配列の要素を文字列にして書き出す
<p>
<script type="text/javascript">
<!--
hairetu= new Array("0番目","1番目","2番目","3番目");
document.write(hairetu.join("/"));
//-->
</script>
</p>
</body>
</html>
```

# 配列の要素を逆に並べ替える

**array オブジェクト名 = new Array(0 番目の要素, 1 番目の要素, ..., n 番目の要素)**
**オブジェクト名.reverse()** ［メソッド］






「reverse()」メソッドは、配列の要素を逆に並べ替えます。
「reverse()」メソッドは、Netscape Navigator 2.X では使用できません。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
  ～中略～
</head>
<body>
*配列の要素を逆に並べ替える
<p>
<script type="text/javascript">
<!--
hairetu= new Array("0番目","1番目","2番目","3番目");
document.write(hairetu.reverse());
//-->
</script>
</p>
</body>
</html>
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

# 配列の要素をソートする

**array オブジェクト名 = new Array(0 番目の要素, 1 番目の要素, ..., n 番目の要素)**
**オブジェクト名.sort(比較関数)** [メソッド]




「sort()」メソッドは、配列の要素をソートします。
比較関数が与えられていない時は、各配列の要素が文字列として扱われ、辞書編集法に従ってソートされます。
また、数値を比較する時は、比較関数を「function 関数名(a, b){ return a - b }」とすると昇順ソートに、「function 関数名(a, b){ return b - a }」とすると降順ソートになります。
「sort()」メソッドは、Netscape Navigator 2.X では使用できません。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
* 配列の要素をソートする

<p>
```

```
<script type="text/javascript">
<!--
Nhairetu1= new Array("location","array","window","form");
document.write("辞書順:",Nhairetu1.sort());
//-->
</script>
</p>

<p>
<script type="text/javascript">
<!--
Nhairetu2= new Array(62,8,256,32);
function SortSet(a,b) { return a - b }
document.write("昇順:",Nhairetu2.sort(SortSet));
//-->
</script>
</p>

<p>
<script type="text/javascript">
<!--
Nhairetu2= new Array(62,8,256,32);
function SortSet(a,b) { return b - a }
document.write("降順:",Nhairetu2.sort(SortSet));
//-->
</script>
</p>

</body>
</html>
```



**注意**

### 「Sort()」メソッドを使用する時の注意

「Sort()」メソッドを使って配列をソートした結果を、連続して「document.write」で書き出そうとすると、一部あるいは全部が表示されない場合があります。

これは、配列を評価してソートするのに時間がかかり、結果を返すより前にページのロードが終ってしまうためだと思われます。

マシンパワーにも左右されるようなので、「Sort()」メソッドを使う時には、色々な環境でチェックすることをお勧めします。

この問題は、Netscape Navigator 4.0では解消されています。

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

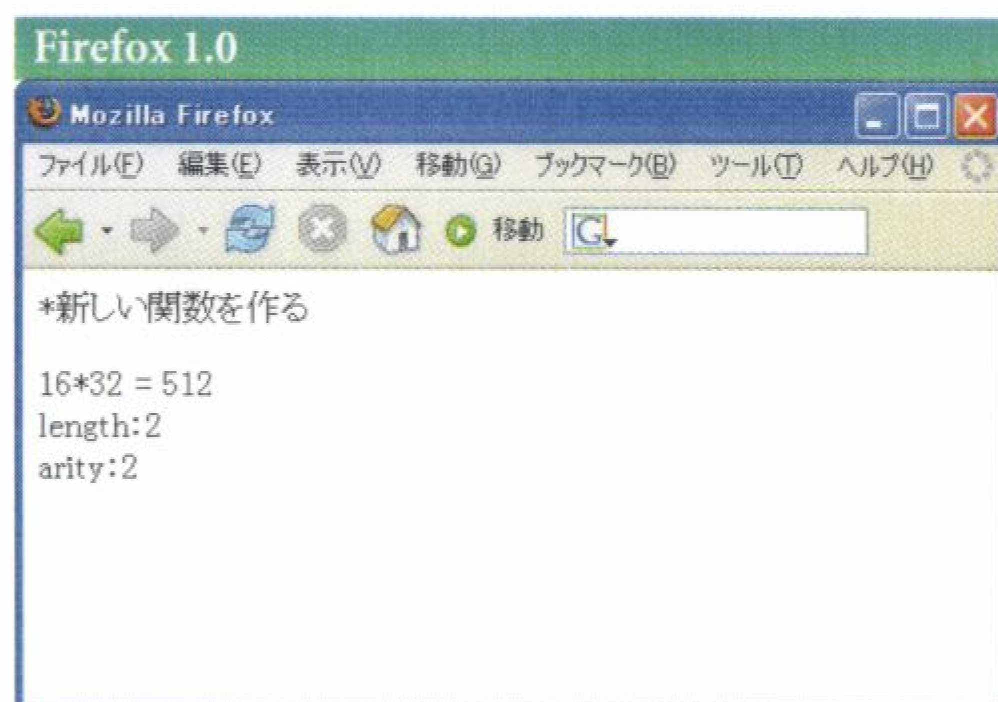
# 新しい関数を作る

オブジェクト名= new Function ([要素1,要素2,...,要素n], function の働き)
オブジェクト名.length [プロパティ]
オブジェクト名.arity [プロパティ]

Internet Explorer 6.0




Firefox 1.0




サンプルでは、new演算子を使って「x」と「y」のふたつの要素を乗算する関数「myFunc」を作り、そこに数値を入れて計算しています。
また、「length」プロパティと「arity」プロパティは、要素数を値に持っています。
JavaScript1.1で追加されたオブジェクトですが、「length」プロパティはJavaScript1.1、「arity」プロパティはJavaScript1.2で追加されました。
Netscape系のブラウザ以外では、「arity」プロパティは機能しません。

## Sample

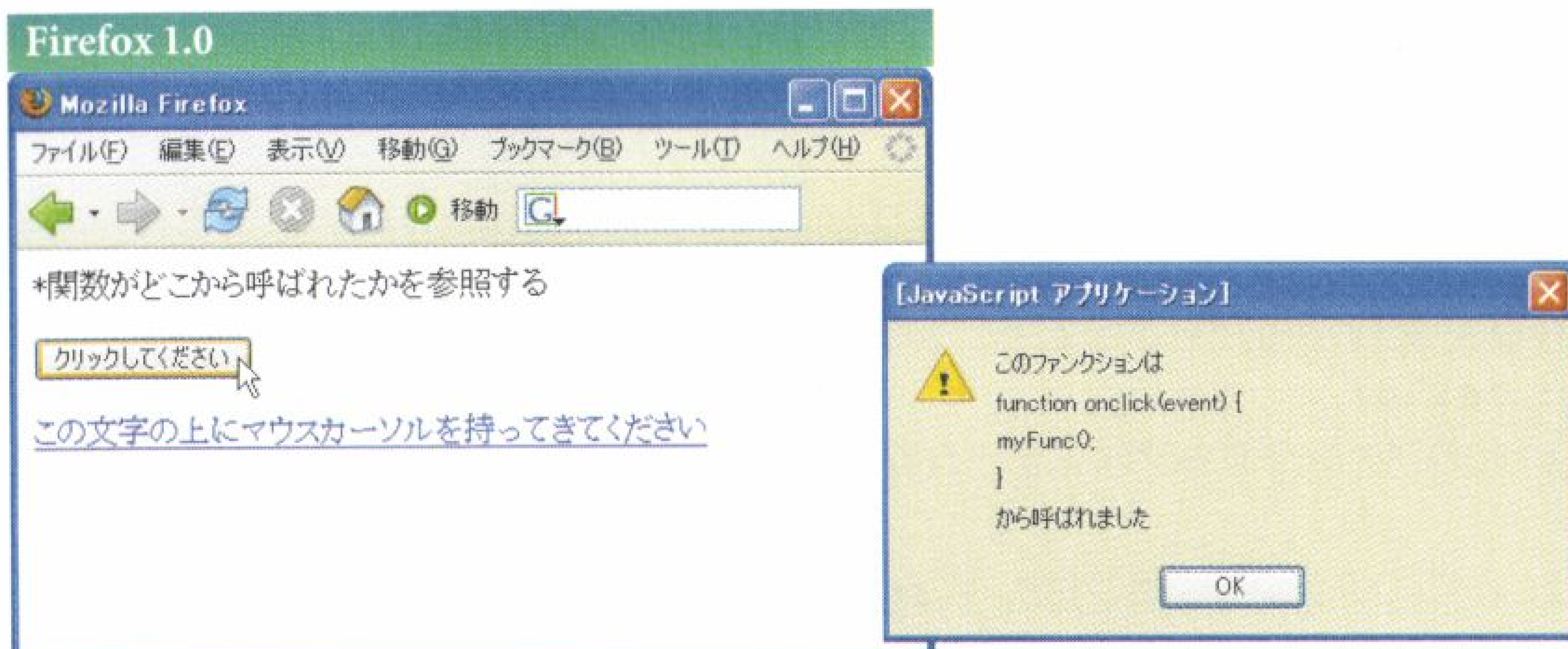
```
<p>
<script type="text/javascript">
<!--
myFunc = new Function("x", "y", "return x * y");
document.write("16*32 = " + myFunc(16,32));
document.write("<br>");
document.write("length:" + myFunc.length);
//-->
</script>
<script type="text/javascript">
<!--
document.write("<br>");
document.write("arity:" + myFunc.arity);
//-->
</script>
</p>
```

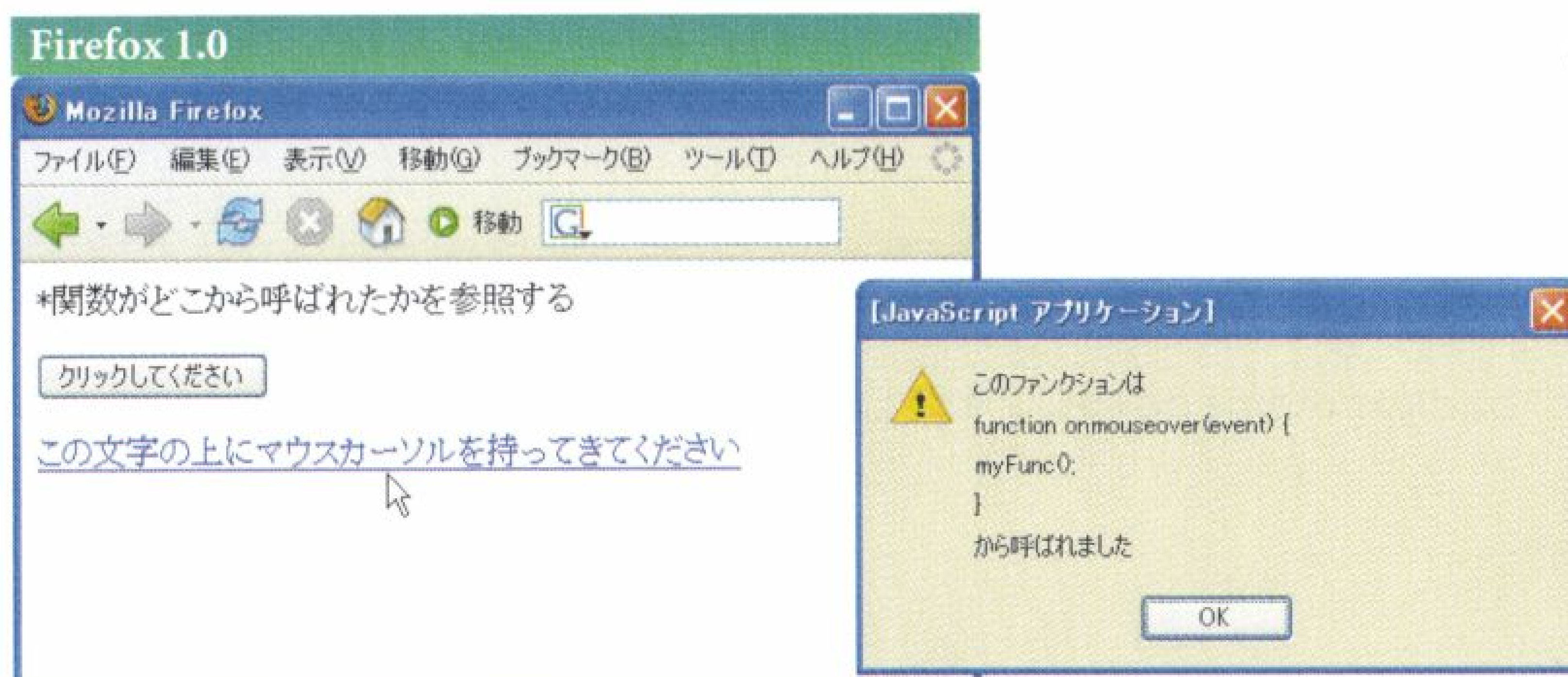
関数オブジェクトを操作する

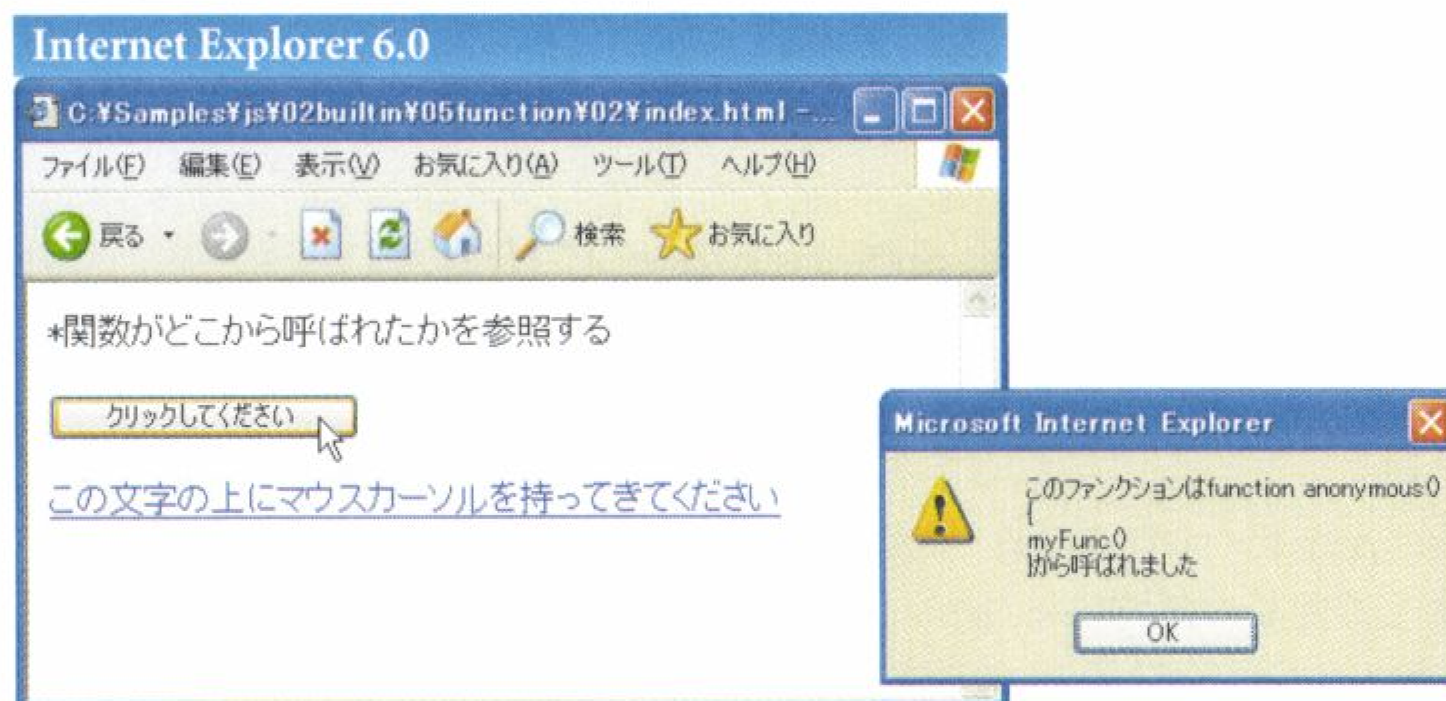
# 関数がどこから呼ばれたかを参照する

**オブジェクト名.caller** ［プロパティ］






「caller」プロパティは、どの関数がどこから呼ばれたかを取得します。
サンプルでは、フォームのボタンをクリックしたり、リンクの上にマウスカーソルが乗った時などに発生した関数名と共に、関数を発生させたイベントのタイプを表示した警告用のダイアログボックスを開きます。ただし、Windows版のInternet Explorerでは、イベントタイプが「anonymous()」となってしまいます。
同様の働きをするプロパティに、オブジェクトの作成元、つまり関数名を取得する「constructor」プロパティがあります。
JavaScript1.1で追加されたオブジェクトです。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function myFunc() {
    if (myFunc.caller == null) {
        alert("このファンクションはフォームのTopから呼ばれました");
    }
   else { alert("このファンクションは" + myFunc.caller + "から呼ばれました");
   }
}
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*関数がどこから呼ばれたかを参照する
<p>
<form>
<input type="button" value=" クリックしてください " onclick="myFunc()">
</form>
</p>
<p>
<a href="PT.html" onMouseOver="myFunc()">この文字の上にマウスカーソルを
持ってきてください</a>
</p>
</body>
</html>
```


alert()：「windowオブジェクト」の「警告用のダイアログボックスを開く」(P.333)

# 関数の内容を配列として使用する

**オブジェクト(ファンクション)名.arguments.length** [プロパティ]
**オブジェクト(ファンクション)名.arguments.callee** [プロパティ]
**オブジェクト(ファンクション)名.arguments[インデックス]**






「arguments」は、それ自体が関数の配列を作成するオブジェクトで、「length」プロパティを持っています。JavaScript1.0に対応しているNetscape Navigator 2.0から使用可能でしたが、JavaScript1.1に対応したNetscape Navigator 3.0からは、functionオブジェクトのプロパティに加えられました。
サンプルでは、「list.arguments.length」で関数「list()」の要素数を取得し、for文を使って「list.arguments[0]」の要素から順番に書き出しています。また、「callee」プロパティは、関数の内容を値に持っています。
JavaScript1.1で追加されたオブジェクトですが、「length」プロパティはJavaScript1.1、「callee」プロパティはJavaScript1.2で追加されました。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function list() {
    for (var i=0; i<list.arguments.length; i++) {
        document.write("<li>" + list.arguments[i]);
    }
document.write("<br>");
document.write("関数の内容:" + list.arguments.callee);
}
//-->
</script>
```

```
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*関数の内容を配列として使用する
<p>
<script type="text/javascript">
<!--
list("One","Two","Three")
//-->
</script>
</p>
</body>
</html>
```

**注意**

## ポップアップ ブロックに関する注意

JavaScriptでポップアップ ブロックに引っ掛かるのは、「window.open()」メソッドを使って自動的に新しいウィンドウを開いた時です。

たとえば、本書の場合、「ページを抜ける時に新しいウィンドウを開く」(P.342)や「リンクの上にポインタが乗るとウィンドウを開く」(P.419)を実行した場合は、ポップアップ ブロックに引っ掛かります。ただし、「window.open()」メソッドを使ったとしても、「新しいウィンドウを開く」(P.338)のように、ボタンのクリックなどのユーザの操作によってウィンドウを開いた場合には、ポップアップ ブロックには引っ掛かりません。また、「window.alert()」「confirm()」「prompt()」の各メソッドは、ポップアップ ブロックの対象にはなりません。

以上のことから、イベントハンドラ「onLoad」や「onUnload」「onMouseOver」などを使って「window.open()」メソッドを呼び出すような時は、ポップアップ ブロックに引っ掛かってしまうため、サイトを構築する場合には注意が必要です。

ポップアップ ブロックも含めて、Windows XP Service Pack 2をあてた後のInternet Explorer 6を対象にしたサイトを構築する上での注意点については「Windows XP Service Pack 2への対応に向けたWebサイトの最適化」を、ポップアップ ブロックについては「Windows XP SP2の新機能、ポップアップ ブロックの概要」を参考にするとよいでしょう。

**「Windows XP Service Pack 2への対応に向けたWebサイトの最適化」**

(http://www.microsoft.com/japan/msdn/windows/windowsxp/xpsp2web.asp)

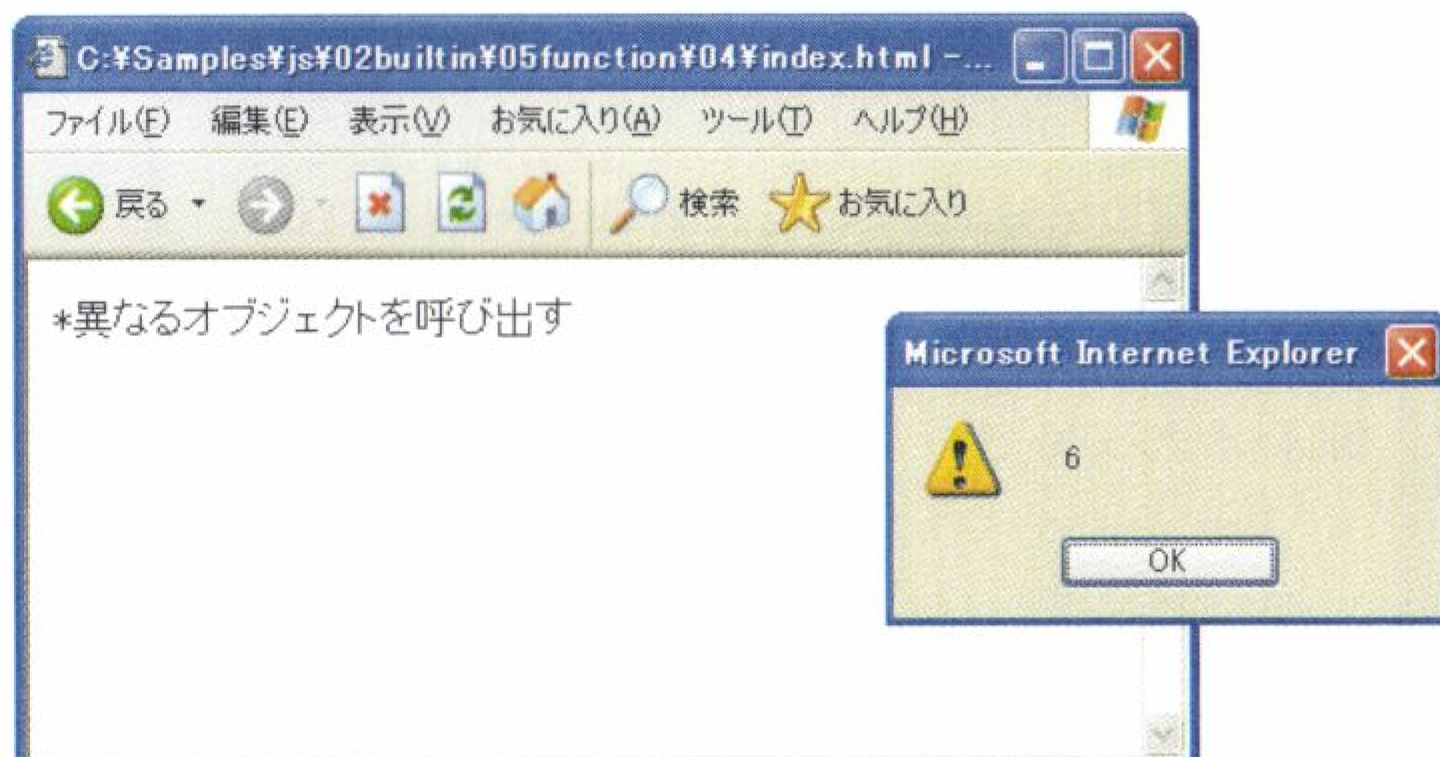
**「Windows XP SP2の新機能、ポップアップ ブロックの概要」**

(http://support.microsoft.com/kb/883735/JA/)

# 異なるオブジェクトを呼び出す

**call(this, 引数,引数,...)** [メソッド]





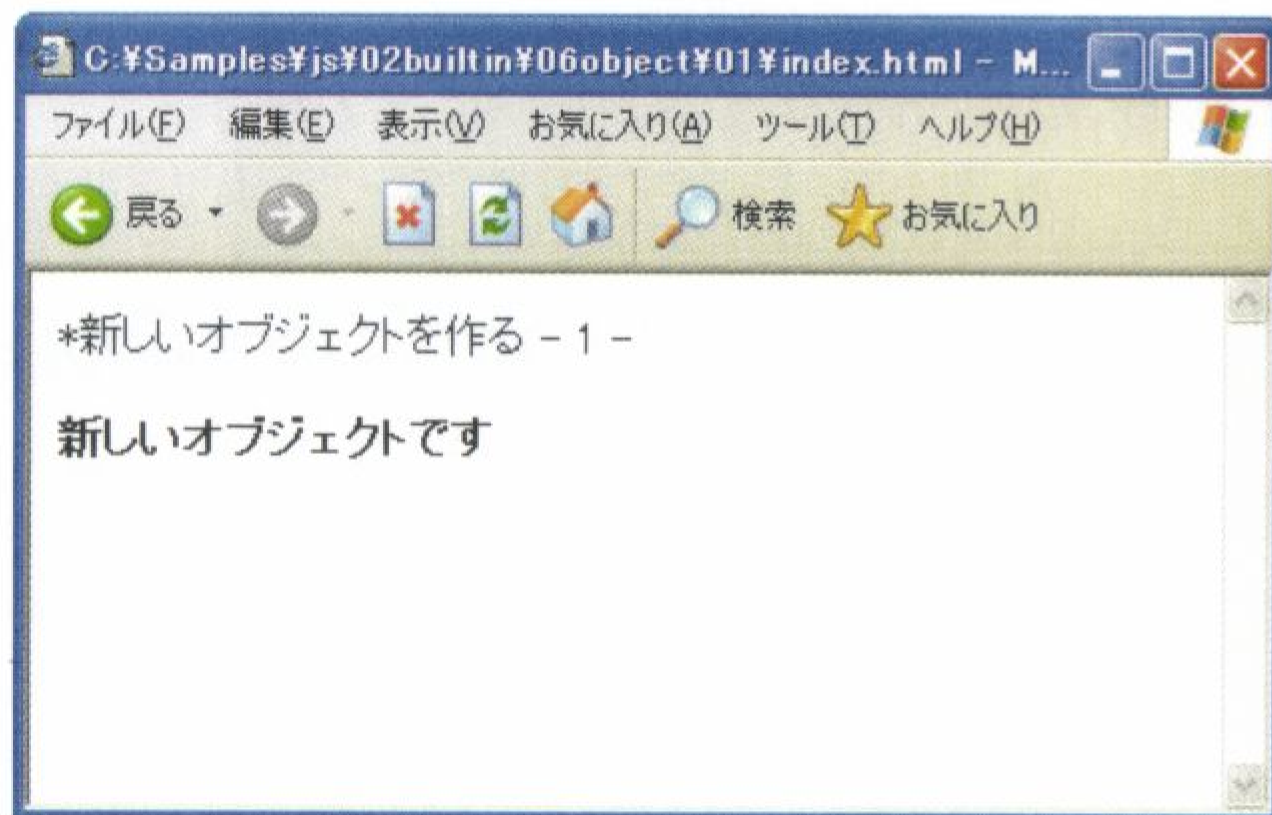

「call()」メソッドは、引数を指定して、異なるオブジェクトを呼び出します。
サンプルでは、「x」と「y」のふたつの要素を乗算する関数であるオブジェクト「myFunc」を、関数「myFunc2()」から「call()」メソッドを使って呼び出しています。
この他にも、引数の配列と共にオブジェクトを呼び出す「apply()」メソッドがあり、両方ともJavaScript1.3で追加されたメソッドです。
このメソッドは、Internet Explorer 5.0ではサポートされていませんが、Internet Explorer 5.5と6.0ではサポートされています。

## Sample

```
<script type="text/javascript">
<!--
myFunc = new Function("x", "y", "return x * y");
function myFunc2(a,b) { window.alert( myFunc.call(this, a, b) )
}
//-->
</script>
 ～中略～
<body>
*異なるオブジェクトを呼び出す
<p>
<script type="text/javascript">
<!--
myFunc2(2,3)
//-->
</script>
</p>
</body>
```

# 新しいオブジェクトを作る - 1 -

**オブジェクト名 = new Object()**




JavaScriptでは、新たにユーザー独自のオブジェクトを作成することができます。
サンプルでは、「new」演算子を使って、「新しいオブジェクトです」という文字列の値を持った新たなオブジェクト、「myObj」を作成しています。
JavaScript1.0からのオブジェクトですが、Netscape Navigator 2.0ではうまく機能しない場合があります。

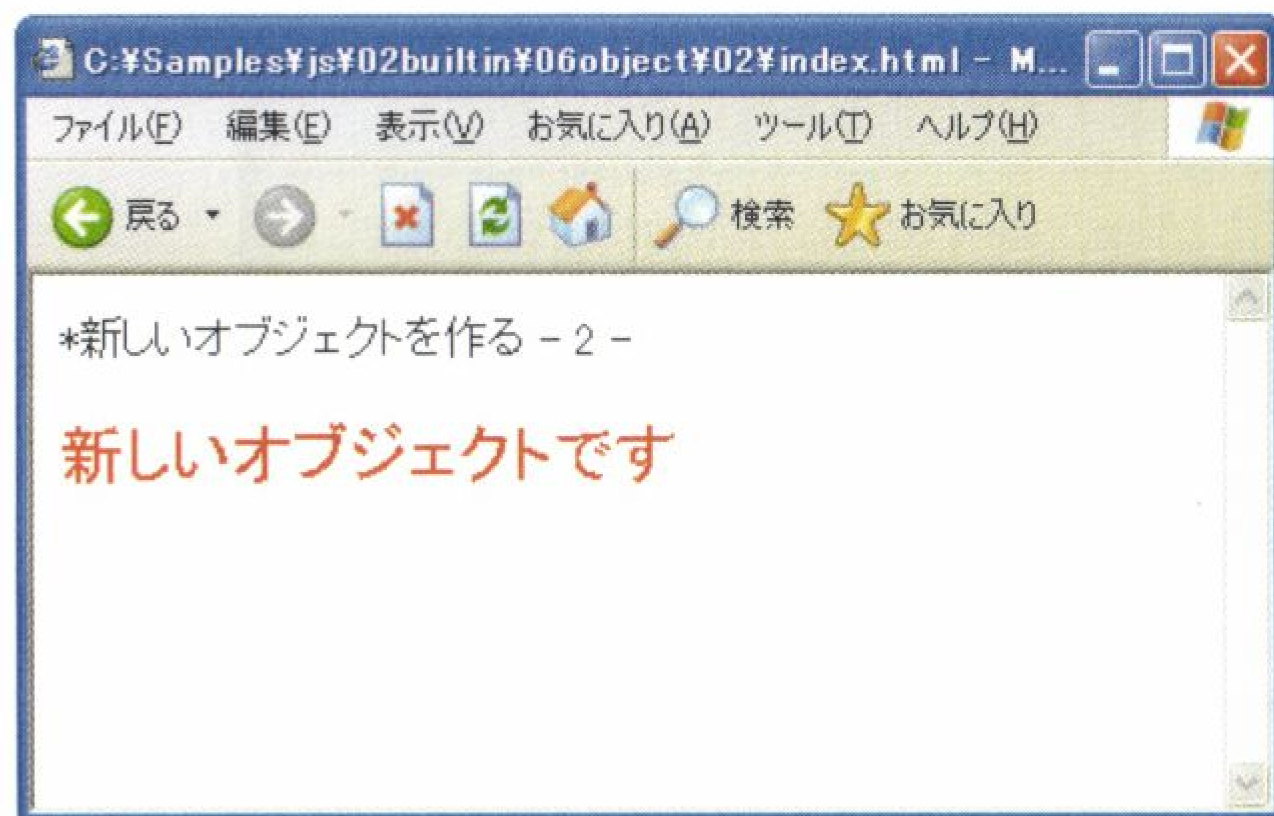
## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*新しいオブジェクトを作る - 1 -
<p>
<script type="text/javascript">
<!--
myObj = new Object("新しいオブジェクトです");
document.write(myObj.bold());
//-->
</script>
</p>
</body></html>
```

# 新しいオブジェクトを作る - 2 -

**オブジェクト名 = {プロパティ 1:値 1,プロパティ 2:値 2,..., プロパティ n:値 n}**






JavaScript1.2からは、あらかじめ用意されているオブジェクトを使用するか、new演算子を使用して新しいオブジェクトを作成する以外にも、サンプルの用法でも独自のオブジェクトを作成できるようになりました。
サンプルでは、文字の色やサイズの値をプロパティに持った「mystring」というオブジェクトを作成しています。

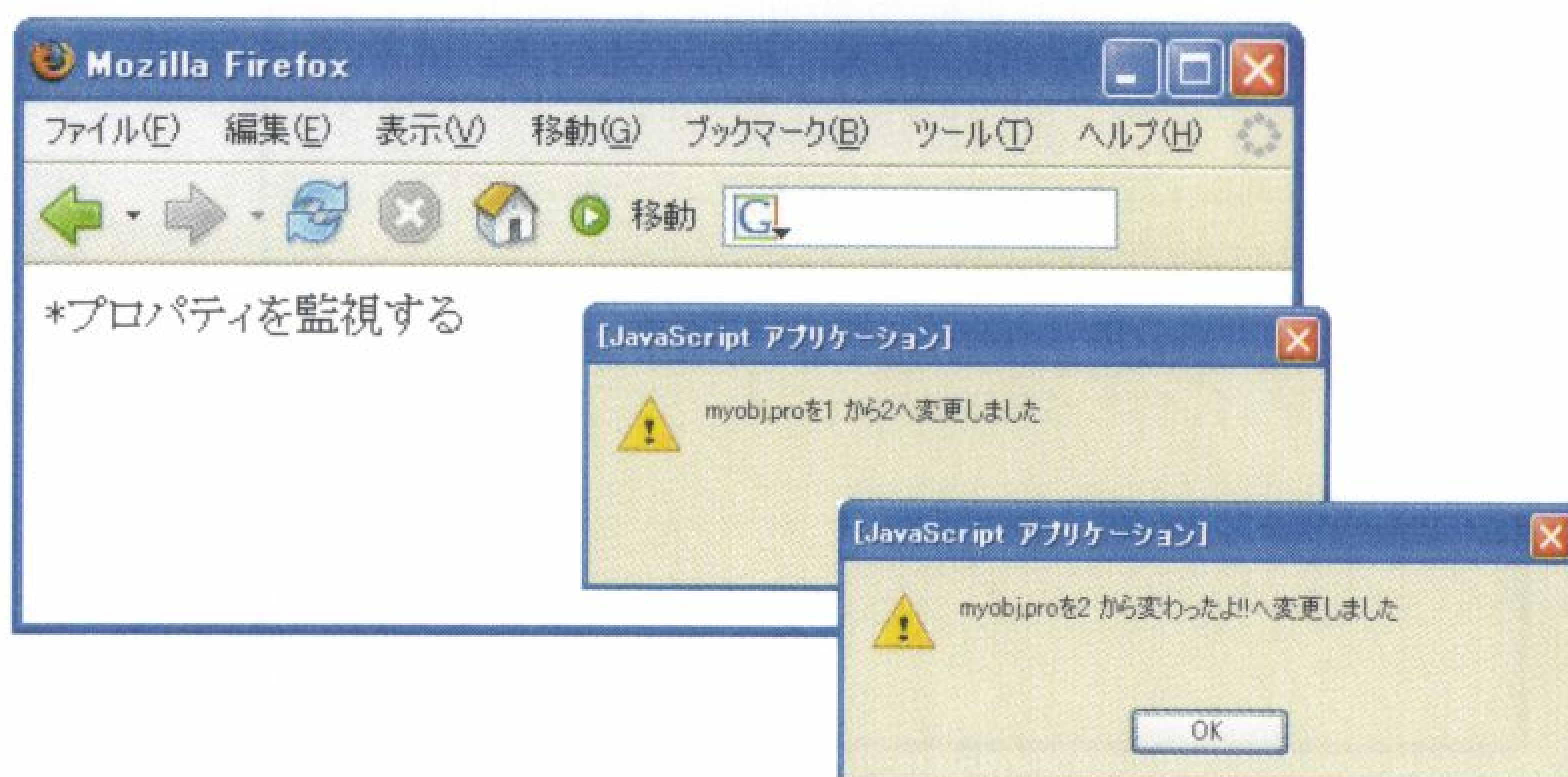
## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
  〜中略〜
</head>
<body>
*新しいオブジェクトを作る - 2 -
<p>
<script type="text/javascript">
<!--
myObj = new Object("新しいオブジェクトです");
mystring = { color:"red",size:5 }
document.write(myObj.fontcolor(mystring.color).fontsize(mystring.siz
e));
//-->
</script>
</p>
</body>
</html>
```

# プロパティを監視する

**watch(プロパティ,関数)** [メソッド]
**unwatch(プロパティ)** [メソッド]






「watch()」メソッドは、プロパティの監視を行うメソッドです。
サンプルでは、新たに作成した「myobj」オブジェクトの「pro」プロパティを「watch()」メソッドで監視しています。「pro」プロパティには、始め「1」の値が設定されていますが、「watch()」メソッド内で呼び出す関数の処理で「1」から「2」へ、「2」から「変わったよ!!」という文字列へと変更されていきます。ちなみに、この時の関数の引数は、「function (プロパティ,変更前の値,変更後の値)」となります。
「unwatch()」メソッドは、「watch()」メソッドで設定したプロパティを監視を解除するメソッドです。このため、「myobj.unwatch('pro')」の後に設定した「myobj.pro = 100」の処理は行われません。
JavaScript1.3で追加されたメソッドですが、Internet Explorer 5.Xと6.Xではサポートされていないようです。

## Sample

```
<script type="text/javascript">
<!--
myobj = {pro:1}
myobj.watch("pro",
   function (id,oldval,newval) {
      window.alert("myobj." + id + "を"  + oldval + " から" + new
val + "へ変更しました");
      return newval;
   })
myobj.pro = 2;
myobj.pro = "変わったよ!!";
myobj.unwatch('pro');
myobj.pro = 100;
//-->
</script>
```

# 真(true)か偽(false)の値を設定する

オブジェクト名 = new Boolean()






Booleanオブジェクトを使うと、明示的に「true」と「false」の値を持ったオブジェクトを作成することができます。
サンプルでは、Booleanオブジェクトを使って「true」の値を持った「myTrue」オブジェクトと、「false」の値を持った「myFalse」オブジェクトを作成しています。
JavaScript1.1で追加されたオブジェクトです。

## Sample

```
<script type="text/javascript">
<!--
myTrue = new Boolean(true);
myFalse = new Boolean(false);
document.write( myTrue == true );
document.write("<br>");
document.write( myTrue == false );
document.write("<br>");
document.write( myFalse != true );
document.write("<br>");
document.write( myFalse != false );
//-->
</script>
```

IE6.0
IE5.5
IE5.0
IE4.0
Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X
Opera7
Opera6
Safari
IE5-mac
IE4-mac

# 数値を作成する

**オブジェクト名 = new Number()**
**オブジェクト名.NaN** [プロパティ]




Numberオブジェクトを使うと、数値の値を持ったオブジェクトを作成することができます。
サンプルでは、Numberオブジェクトを使って、数値の値を持ったオブジェクト「myNumber」を作成しています。また、「NaN」プロパティを使うと、オブジェクトを数値でない状態に変更することができます。
JavaScript1.1で追加されたオブジェクトですが、Netscape Navigator 3.Xの一部では動作しない場合があります。

## Sample

```
<script type="text/javascript">
<!--
myNumber = new Number(100);
document.write(myNumber);
document.write("<br>");
document.write(isNaN(myNumber));
document.write("<br>");
document.write(isNaN(myNumber.NaN));
//-->
</script>
```

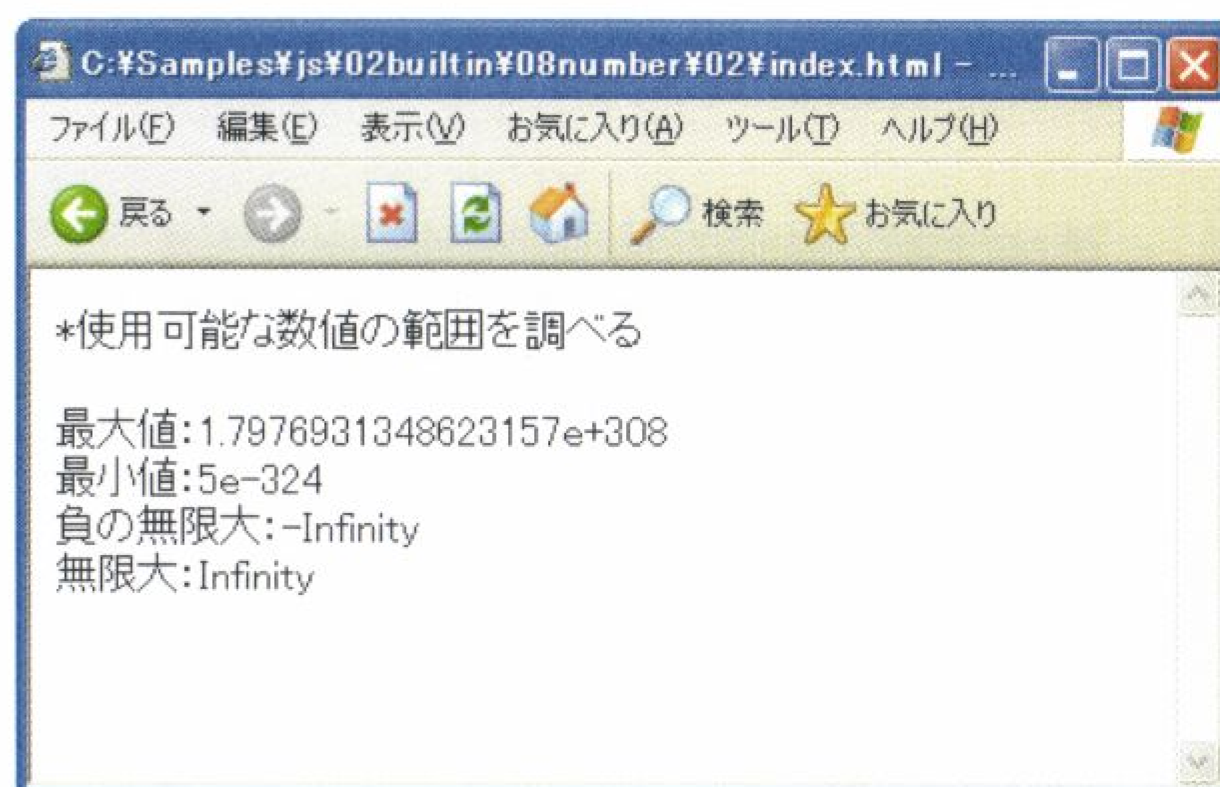
関数オブジェクトを操作する


isNaN():「ビルトイン関数(top-level関数)」の「数値かどうかを調べる」(P.599)

# 使用可能な数値の範囲を調べる

オブジェクト名.**MAX_VALUE** [プロパティ]
オブジェクト名.**MIN_VALUE** [プロパティ]
オブジェクト名.**NEGATIVE_INFINITY** [プロパティ]
オブジェクト名.**POSITIVE_INFINITY** [プロパティ]






Numberオブジェクトには、数値の有効範囲を取り扱う多くのプロパティがあります。「MAX_VALUE」プロパティはJavaScriptで扱える値の最大値を、「MIN_VALUE」プロパティはJavaScriptで扱える値の最小値を、「NEGATIVE_INFINITY」プロパティは負の無限大の値を、「POSITIVE_INFINITY」プロパティは無限大の値を、それぞれ持っています。

## Sample

```
<script type="text/javascript">
<!--
document.write("最大値:" + Number.MAX_VALUE);
document.write("<br>");
document.write("最小値:" + Number.MIN_VALUE);
document.write("<br>");
document.write("負の無限大:" + Number.NEGATIVE_INFINITY);
document.write("<br>");
document.write("無限大:" + Number.POSITIVE_INFINITY);
//-->
</script>
```

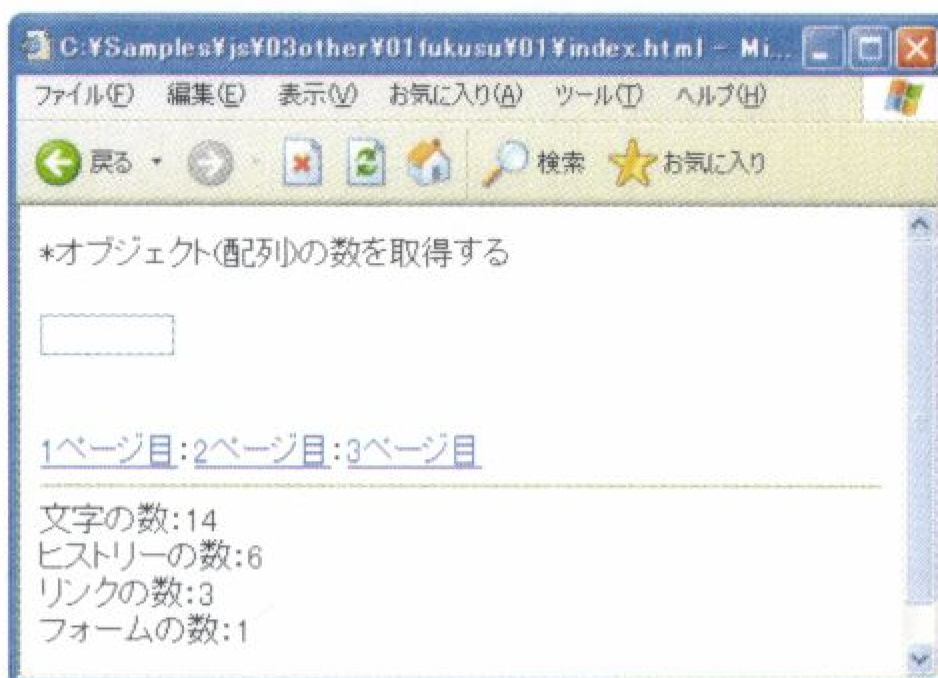
IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# オブジェクト(配列)の数を取得する

## length [プロパティ]

【サポートしているオブジェクト】
Button, Checkbox, FileUpload, Form, Frame, Hidden, Image, Layer, Password, Plugin, Radio, Reset, Select, Submit, Text, Textarea, window

【サポートしている配列】
anchors, applets, elements(フォーム内の値), embeds, forms, frames, history, images, layers, links, mineTypes, options, plugins




プロパティやメソッドの中には、複数あるいはすべてのオブジェクトで使用できるものがあります。
「length」プロパティは、HTMLファイル内のオブジェクトや配列の数を取得します。実際に試す場合には、この他にも「page1.html」~「page3.html」の3つのHTMLファイルを用意してください。

## Sample

```
<form name="kasu1">
<input type="text" name="X" size="10">
</form>
<br>
<a href="page1.html">1ページ目</a>:<a href="page2.html">2ページ目</a>
:<a href="page3.html">3ページ目</a>
<hr>
<script type="text/javascript">
<!--
var mozi = "この文字は何文字でしょうか";
    document.write("文字の数:",mozi.length,"<br>");
    document.write("ヒストリーの数:",history.length,"<br>");
    document.write("リンクの数:",document.links.length,"<br>");
    document.write("フォームの数:",document.forms.length,"<br>");
//-->
</script>
```

複数で利用する

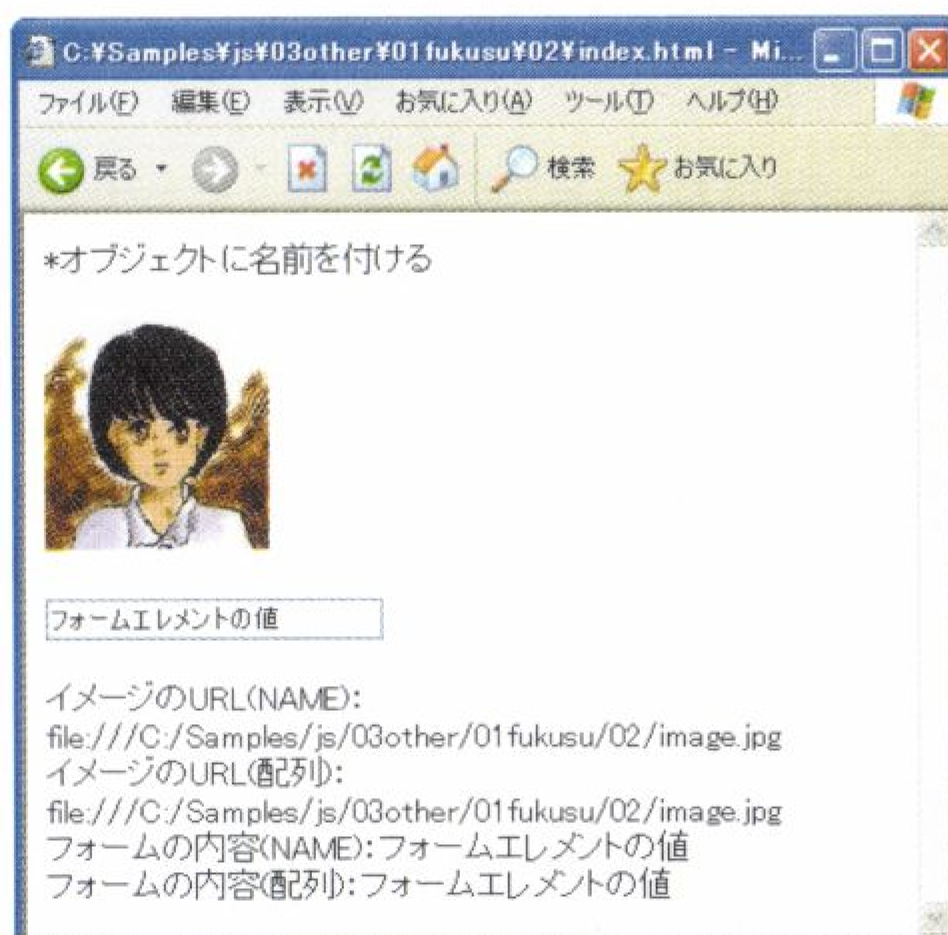
# オブジェクトに名前を付ける

## name [プロパティ]

【サポートしているオブジェクト】

Button, Checkbox, FileUpload, Form, Frame, Hidden, Image, Layer, Password, Plugin, Radio, Reset, Select, Submit, Text, Textarea, window






Link関連のオブジェクト以外のHTMLタグを使用するオブジェクトでは、HTMLタグ内に「name」プロパティを設定することによって、オブジェクトに名前を付けて、明示的にオブジェクトを参照することができるようになります。
サンプルでは、Imageオブジェクトと Formオブジェクトに「name」プロパティでオブジェクトに名前を設定することによって、その名前で配列と同じようにオブジェクトの色々な値を参照しています。

### Sample

```
<p>
<img src="image.jpg" name="kao" alt="kao" width="110" height="110">
</p>
<p>
<form name="NAIYOU">
<input type="text" name="naiyou" size="30" value="フォームエレメントの
値">
</form>
</p>

<script type="text/javascript">
<!--
document.write("イメージのURL(NAME):");
document.write(document.kao.src);
document.write("<br>");
```

```
document.write("イメージのURL(配列):");
document.write(document.images[0].src);
document.write("<br>");
document.write("フォームの内容(NAME):");
document.write(document.NAIYOU.naiyou.value);
document.write("<br>");
document.write("フォームの内容(配列):");
document.write(document.forms[0].elements[0].value);
//-->
</script>
```

## コラム

### これからのNetscape

Netscapeの開発を行っているAOLは、2004年11月30日に次期バージョンのプロトタイプを開発者向けに発表し、さらに2005年3月には一般向けのベータ版を公開しました。

それまでは、「AOLはNetscapeの開発をやめるのではないか」との憶測も流れていて、Netscapeの将来を危ぶむ声も出ていました。しかしこれで、AOLが引き続きNetscapeの開発を続けていく意志があると解釈できますし、その意味でも安心している人も多いのではないでしょうか。

今回発表されたNetscapeでは、ベースがMozillaからFirefoxがとなりました。また、次期バージョンの最大の特徴は、Netscape標準のレンダリングエンジンであるGeckoと、Windowsに含まれているInternet Explorerのエンジンとの2種類を切り替えられることにあります。

現在、もっともシェアの高いブラウザは、Internet Explorerです。そのため、Webの本来の理念からは大きく外れてしまうことを妥協して、Internet Explorerでしかまともに見ることができないサイトが数多くあることも事実です。しかし、今回のNetscapeがとった対応により、「通常はGeckoを使ってサイトを表示して、Internet Explorer用に独自に作られたサイトはInternet Explorerに切り替えて表示する」といった運用が可能になります。

もちろん、Windows内に含まれるInternet Explorerのコンポーネントを使用することになるので、このブラウザはWindows版しか発表されておらず、他のOSでどのような対応になるかは、まだ未定です。また、Internet ExplorerがOSと深く結びついているところから来る、Internet Explorer自身のセキュリティ上の問題も、Netscapeに引き継がれてしまう可能性が高いという問題点もあります。

評価は始まったばかりなので、現時点では、次期Netscapeがどのようなものになるか、まだわかりません。

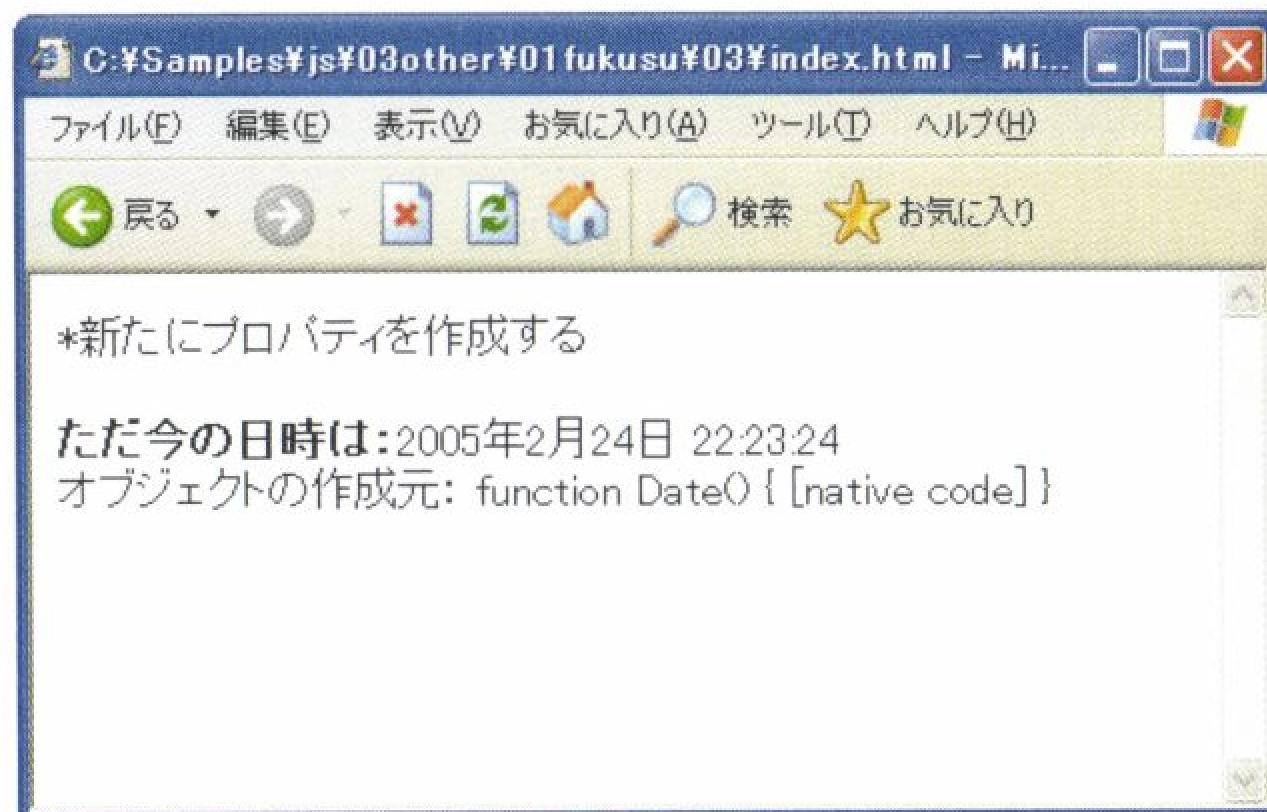
しかし、今でもレンダリングエンジンにGeckoを使ったブラウザや、Internet Explorerのコンポーネントを使った、いわゆる互換ブラウザと呼ばれるブラウザが発表されています。Netscapeの動向によっては、今後はそれらの互換ブラウザでも、複数のエンジンを切り替えできるブラウザがトレンドになっていくかもしれません。

# 新たにプロパティを作成する

**prototype** [プロパティ]
**constructor** [プロパティ]

【サポートしているオブジェクト】
Array, Boolean, Date, Function, Image, Number ,Select, option, String, ユーザー作成オブジェクト






「prototype」プロパティは、new演算子で作られたオブジェクトに、明示的にプロパティを追加します。
サンプルでは、new演算子で作られたtodayオブジェクトに対して、「prototype」プロパティを使って「newpro」というプロパティを新たに作成しています。
また、「constructor」プロパティは、関数名も含めたオブジェクトの作成元の値を持っています。
あくまでも作成元の情報だけなので、そのオブジェクトがどのような値を持っているかは、「toString()」メソッド(次項の「オブジェクトを文字列に変える」)や、「toSource()」メソッド(P.587の「オブジェクト内の値を文字列にする」)を使う必要があります。
JavaScript1.1で追加されたプロパティです。

## Sample

```
<script type="text/javascript">
<!--
today = new Date();
Date.prototype.newpro="ただ今の日時は:".bold();
document.write(today.newpro + today.toLocaleString());
document.write("<br>");
document.write("オブジェクトの作成元:" + today.constructor);
//-->
</script>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# オブジェクトを文字列に変える

## toString() [メソッド]

【サポートしているオブジェクト】
すべてのオブジェクト



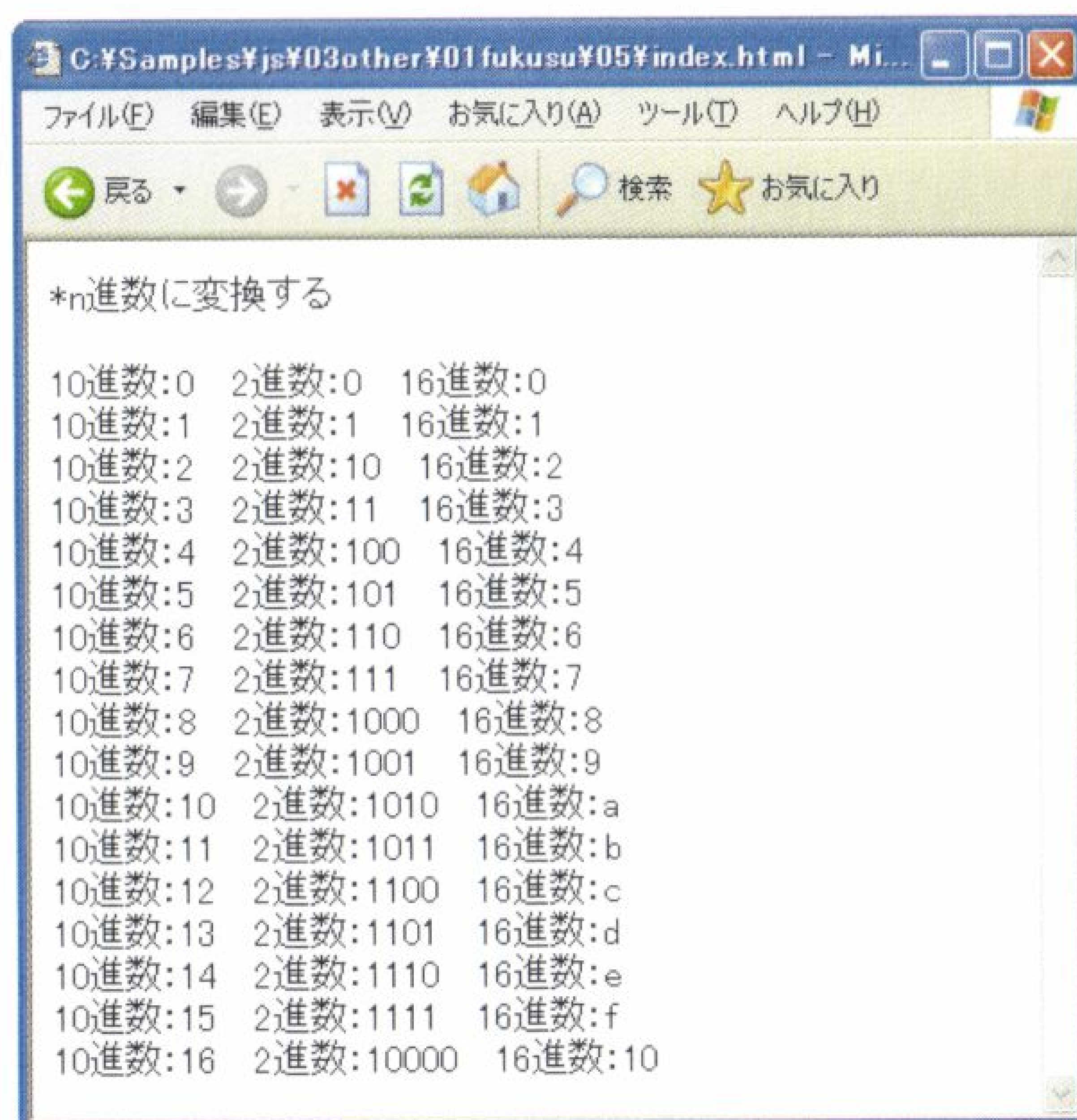

「toString()」メソッドは、オブジェクトを文字列に変換します。
Netscape Navigator 2.0でこのメソッドを使用すると、一部のブラウザでは動作が不安定になる場合があります。

### Sample

```
<script type="text/javascript">
<!--
var YOUBI = new Array("月","火","水","木","金","土");
document.write(YOUBI.toString() + "<br>");
var today = new Date();
document.write(today.toString() + "<br>");
document.write(location.toString() + "<br>");
document.write("文字列にすべき内容がない時:"+window.toString() );
//-->
</script>
```

# n進数に変換する

**toString(*n*)** [メソッド]



「toString()」メソッドで「toString(n)」のように数値を与えると、n進数の数値を返します。

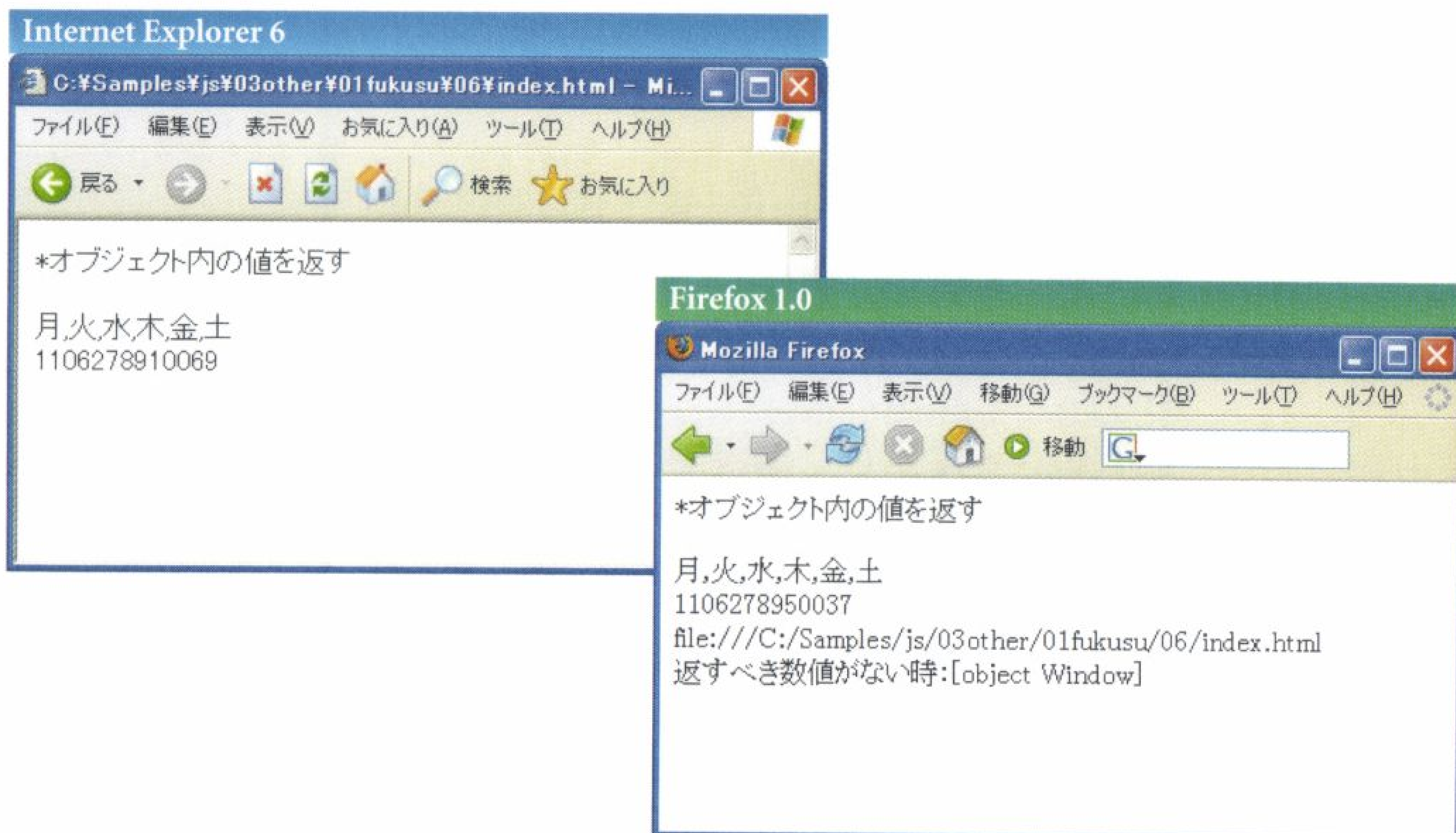
## Sample

```
<script type="text/javascript">
<!--
for (x = 0; x < 17; x++) {
    document.write("10進数:", x.toString(10), "   2進数:", x.toString(2), "   16進数:", x.toString(16),"<br>");
}
//-->
</script>
```

# オブジェクト内の値を返す

**valueOf()** [メソッド]

【サポートしているオブジェクト】
すべてのオブジェクト




「valueOf()」メソッドは、オブジェクト内の値を取得します。
JavaScript1.1で追加されたメソッドです。

## Sample

```
<script type="text/javascript">
<!--
var YOUBI = new Array("月","火","水","木","金","土");
document.write(YOUBI.valueOf() + "<br>");
var today = new Date();
document.write(today.valueOf() + "<br>");
document.write(location.valueOf() + "<br>");
document.write("返すべき数値がない時:"+window.valueOf() + "<br>");
//-->
</script>
```

# オブジェクト内の値を文字列にする

## toSource() [メソッド]

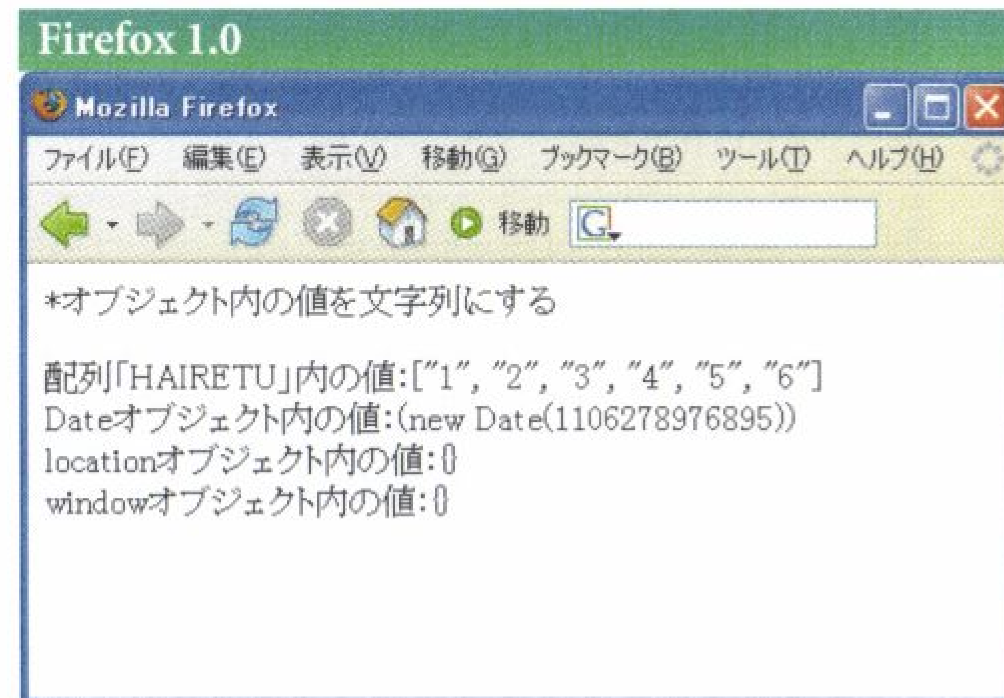
【サポートしているオブジェクト】
すべてのオブジェクト



Internet Explorer 6




Firefox 1.0




「toSource()」メソッドは、オブジェクト内の値を文字列に変換します。
「toString()」メソッドでは「[object Window]」のような形式で表されていた値の詳細も取得することができます。
JavaScript1.3で追加されたメソッドです。
サンプルでは、Netscape 6.Xでナビゲータオブジェクトの値を取得することができません。

### Sample

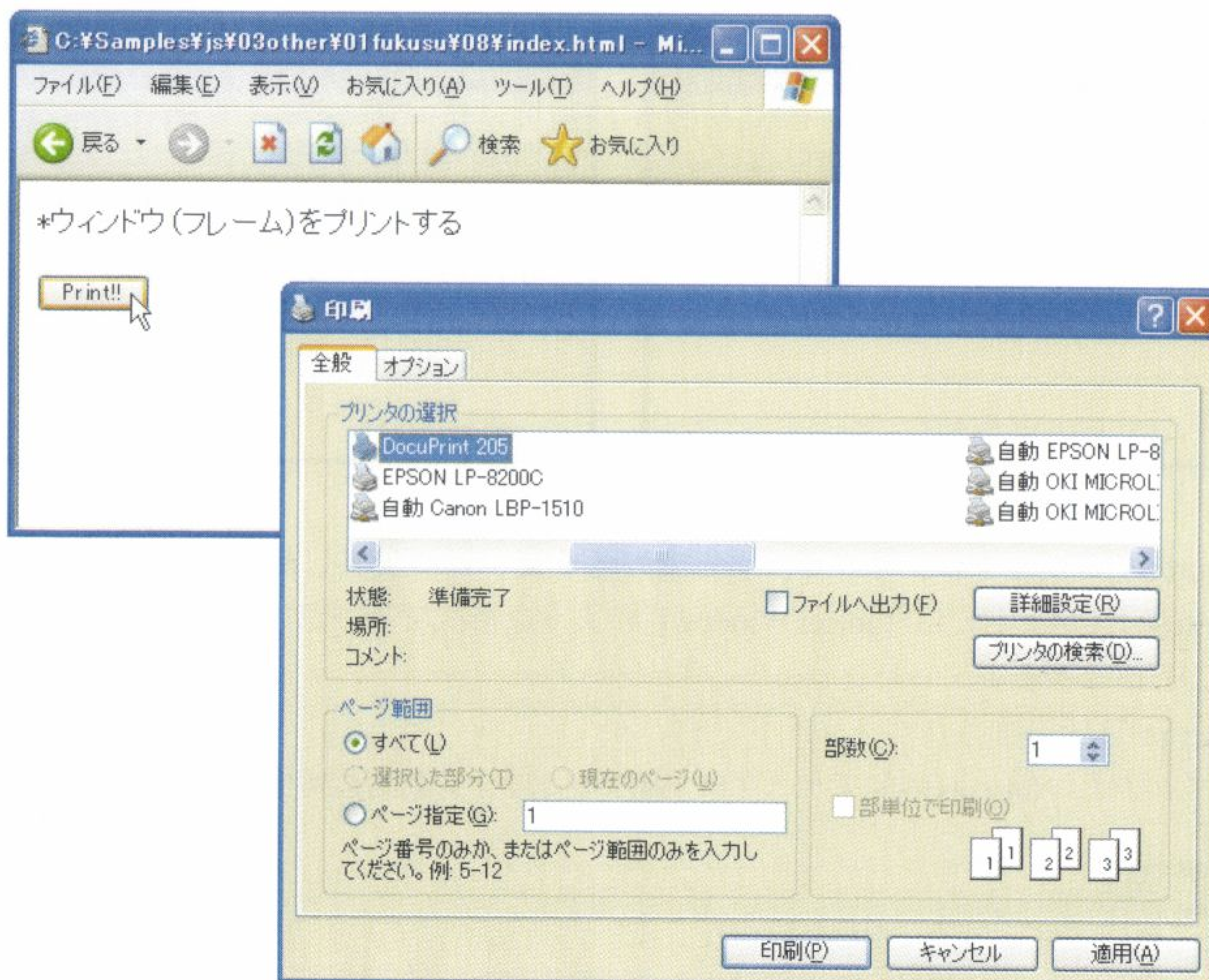
```
<script type="text/javascript">
<!--
var HAIRETU = new Array("1","2","3","4","5","6");
document.write("配列「HAIRETU」内の値:"+HAIRETU.toSource() + "<br>");
var today = new Date();
document.write("Dateオブジェクト内の値:"+today.toSource() + "<br>");
document.write("locationオブジェクト内の値:"+location.toSource() + "<b
r>");
document.write("windowオブジェクト内の値:"+window.toSource());
//-->
</script>
```

JavaScript

# ウィンドウ(フレーム)をプリントする

**オブジェクト名.print()** [メソッド]

【サポートしているオブジェクト】
window, Frame




「print()」メソッドは、指定されているページをプリントします。
サンプルでは、ブラウザのプリントボタンを押すのと同等の働きをするボタンを作成しています。
JavaScript1.2で追加されたメソッドです。現在のところMacintosh版のInternet Explorerでは、サポートされていません。

## Sample

```
<form>
    <input type="button" name="print" value=" Print!! " onclick="window.print()">
</form>
```


<input type="button">:「formオブジェクト」の「ボタンをリンクに使う」(P.420)

# 一定時間ごとに処理を繰り返す

**setInterval(処理,時間設定)** [メソッド]
**clearInterval()** [メソッド]

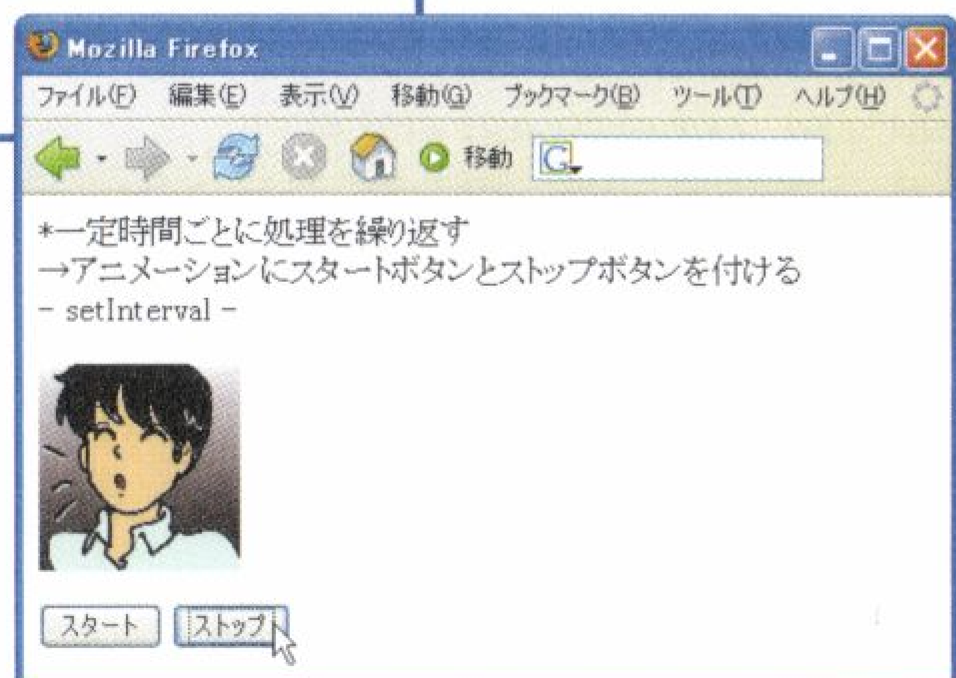
【サポートしているオブジェクト】
window, Frame












「setInterval()」メソッドは、指定した時間後に1度だけ処理を行う「setTimeout()」メソッドとは違い、指定された処理をミリ秒単位で繰り返し行います。
サンプルは、「Imageオブジェクト」の「アニメーションにスタートボタンとストップボタンを付ける」(P.458)を、「setInterval()」メソッドを使って書き直したものです。
「setInterval()」メソッドを停止する時は、「clearInterval()」メソッドを使用して、「setTimeout()」メソッドの用法と同じように「clearInterval(ID名)」とIDを設定して使用します。
JavaScript1.2で追加されたメソッドです。

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Safari IE5-mac IE4-mac

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
var ImageSetA = 1;
ANIMA = new Array();
for(i = 1; i < 6; i++) {
    ANIMA[i] = new Image();
    ANIMA[i].src = "image" + i + ".jpg";
}
function anime1() {
   document.animation.src = ANIMA[ImageSetA].src;
   ImageSetA++;
       if( ImageSetA > 5) {
       ImageSetA = 1;
       }
}
function stop1(){
    clearInterval(IntervarID);
}
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*一定時間ごとに処理を繰り返す<br>
 →アニメーションにスタートボタンとストップボタンを付ける<br>
 - setInterval -
<p>
<img src="image1.jpg" name="animation" alt="Animation" border="0" width="100" height="100">
</p>
<form>
    <input type="button" value=" スタート " onclick="IntervarID=setInterval('anime1()',500)">
    <input type="button" value=" ストップ " onclick="stop1()">
</form>
</body></html>
```

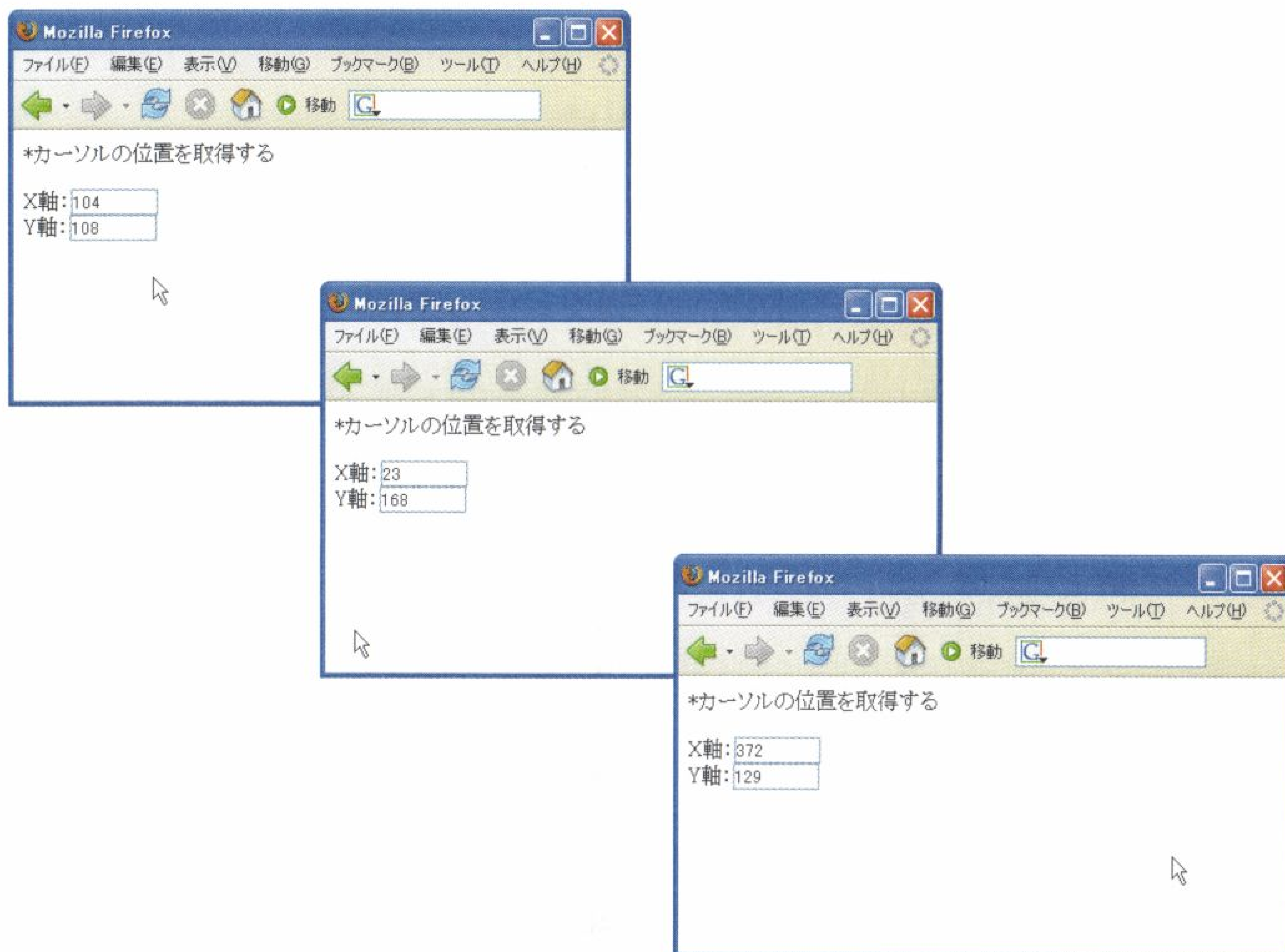
 setTimeout():「windowオブジェクト」の「ステータス行に文字を流す」(P.364)

# カーソルの位置を取得する

**オブジェクト名.captureEvents(Event.イベントタイプ | Event.イベントタイプ | ...)** [メソッド]

【サポートしているオブジェクト】
window, document






「captureEvents()」メソッドは、取得したいイベントを設定することができます。複数のイベントを設定したい時は、「window.captureEvents(Event.イベントタイプ | Event.イベントタイプ | ...)」と記述します。
サンプルでは、「captureEvents()」メソッドで設定した時にのみ使用可能なイベントMouseMoveを設定して、マウスカーソルが移動するたびにカーソルの位置を取得し、フォームに書き出しています。
イベントタイプの記述は、サンプルのように「mousemove」とするか「MOUSEMOVE」と記述してください。「onMouseMove」や「MouseMove」といったように大文字・小文字を混ぜて記述すると、Windows版のNetscape Navigator 4.Xでは動かない場合があります。
JavaScript1.2で追加されたメソッドです。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function Cap(e) {
    document.forms[0].elements[0].value = e.pageX;
    document.forms[0].elements[1].value = e.pageY;
    return true;
}
window.captureEvents(Event.mousemove);
onmousemove=Cap;
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

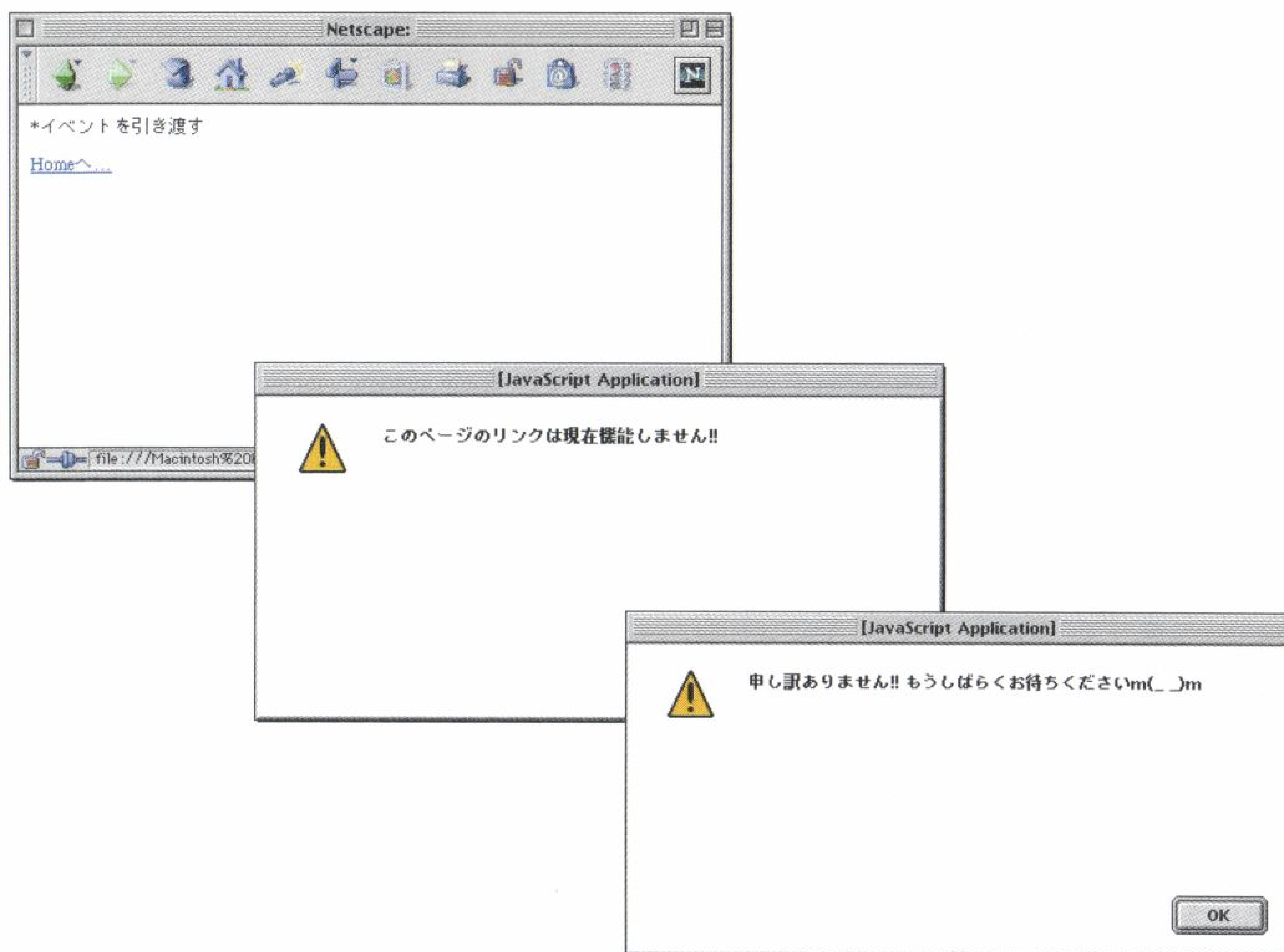
</head>
<body>
*カーソルの位置を取得する
<p>
<form name="ECAP">
X軸：<input type="text" name="ecap" size="10">
<br>
Y軸：<input type="text" name="ecap" size="10">
</form>
</p>
</body>
</html>
```


onmousemove：リファレンス「イベントタイプ」の「MouseMove」(P.623)

# イベントを引き渡す

**オブジェクト名.handleEvent(Event.イベントタイプ)** [メソッド]

**【サポートしているオブジェクト】**
すべてのオブジェクト




「handleEvent()」メソッドは、取得したイベントを他のオブジェクトに引き渡します。
サンプルでは、関数「Cli()」の中で、「window.document.links[0].handleEvent(e)」によってリンク内に設定している「onClick="Cli2()"」にイベントを引き渡し、関数「Cli2()」を発生させています。
この状態では、関数「Cli()」内で設定した警告用のダイアログボックスが開いた後、関数「Cli2()」で設定した警告用のダイアログボックスが開きます。「Cli()」内の「alert()」と「handleEvent()」の設定の順番を逆にすると、「Cli2()」の評価が「Cli()」内で設定した警告用のダイアログボックスが開く前に発生するので、ダイアログボックスの開く順番が逆になります。
イベントタイプの記述は、サンプルのように「click」とするか「CLICK」と記述してください。「onClick」や「Click」といったように大文字・小文字を混ぜて記述すると、Windows版のNetscape Navigator 4.Xでは動かない場合があります。
JavaScript1.2で追加された、イベントを取り扱うすべてのオブジェクトで使用可能なメソッドです。

Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function Cli(e) {
    alert ("このページのリンクは現在機能¥しません!!");
    window.document.links[0].handleEvent(e);
    return false;
        }
function Cli2() {
    alert ("申¥し訳ありません!! もうしばらくお待ちくださいm(_ _)m");
    return true;
}
window.captureEvents(Event.click);
onclick=Cli;
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*イベントを引き渡す
<p>
<a href="../../home.html" onClick="Cli2()">Homeへ...</a>
</p>
</body>
</html>
```

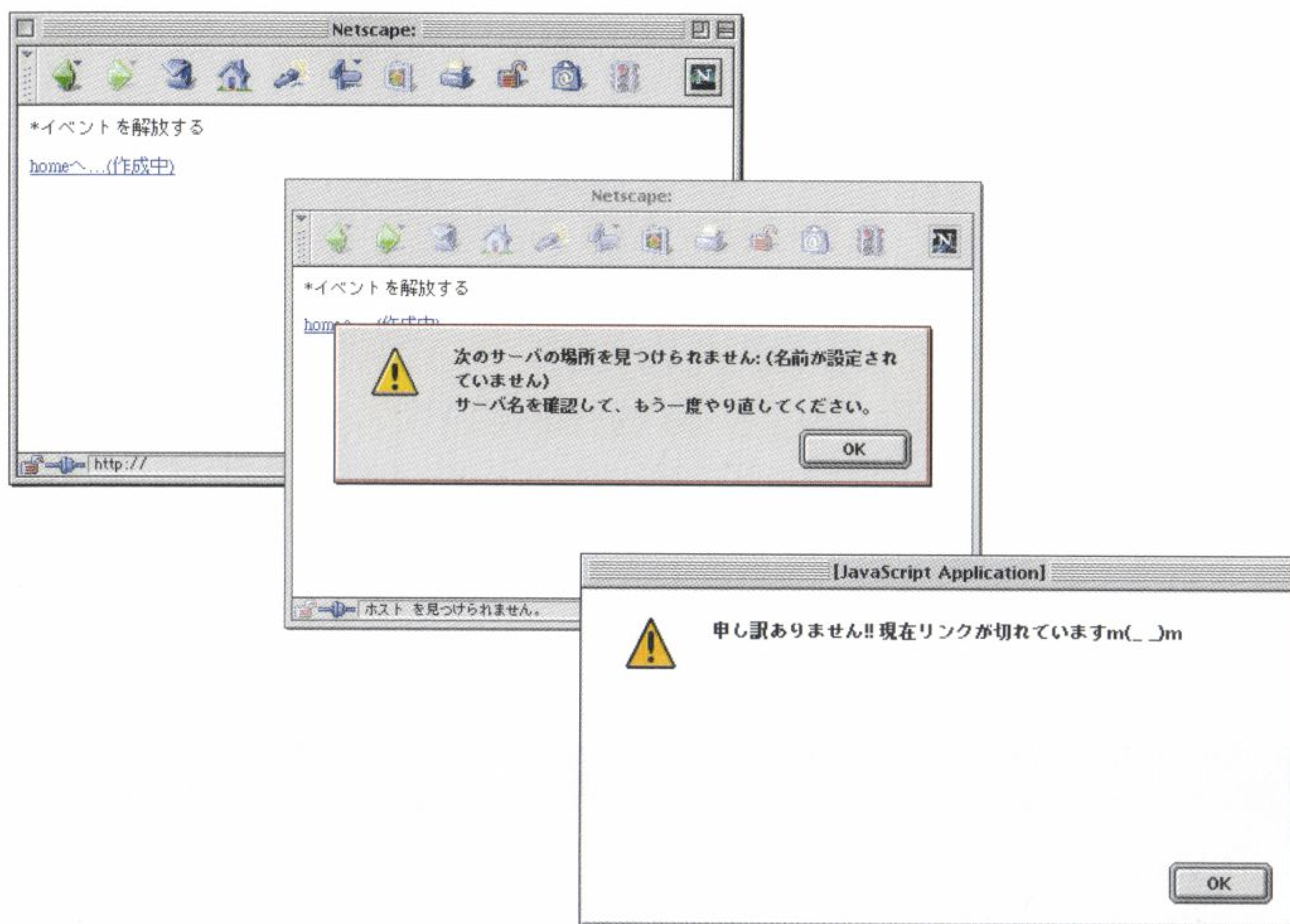

onclick：リファレンス「イベントタイプ」の「Click」(P.622)

# イベントを解放する

**オブジェクト名.releaseEvents(Event.イベントタイプ)** [メソッド]

【サポートしているオブジェクト】
window, document, layer






「releaseEvents()」メソッドは、指定したイベントを解放します。
サンプルでは、リンクの中に「onMouseOut」のイベントを設定しているのですが、このイベントは、1度ウィンドウがクリックされると関数「Cli()」が発生し、関数内で「releaseEvents(Event.MouseOut)」によってイベントが解放されるまでは評価されません。
つまり、リンクがクリックされるまではマウスカーソルがリンクの上に乗っても外れても何も起こらないのですが、1度リンクがクリックされると、それ以降はリンク上からマウスカーソルが外れるとイベントが発生し、警告用のダイアログボックスが開くようになります。
サンプルは、Macintosh版のNetscape Navigator 4.Xでのみ動作が確認されています。
JavaScript1.2で追加されたメソッドです。



## Sample

```
<script type="text/javascript">
<!--
function Cli(e) {
    window.releaseEvents(Event.mouseout);
```

```
    return true;
}
function Mou() {
    alert ("申¥し訳ありません!! 現在リンクが切れていますm(_ _)m");
    return true;
}
window.captureEvents(Event.click);
onclick=Cli
//-->
</script>
  ～中略～
<p>
<a href="http://" onMouseOut="Mou()">homeへ...(作成中)</a>
</p>
```

 onclick：リファレンス「イベントタイプ」の「Click」(P.622)
onMouseOut：リファレンス「イベントタイプ」の「MouseOut」(P.623)

**コラム**

## 「eval()」の変遷

JavaScriptの中で、「eval()」ほど扱いの変更を繰り返したものは、ほかにないでしょう。

元々「eval()」は、JavaScript1.0(Netscape Navigator 2.0～)ではビルトイン関数としてオブジェクトと結び付けられていました。しかし、JavaScript1.1(Netscape Navigator 3.0)からはメソッドになり、それがJavaScript1.2(Netscape Navigator 4.0)からは再びビルトイン関数となりました。

さらに、実際にサポートしたブラウザは発表されませんでしたが、JavaScript1.4からはObjectオブジェクトのメソッドになり、加えて「eval()」メソッドは間接的に呼び出すことはできないということになりました。

さすがに、これで一段落するだろうと思っていたのですが、Netscape 6.0からサポートされているJavaScript1.5のリファレンスでObjectオブジェクトの項目を見てみると、「The eval method is no longer available as a method of Object. Use the top-level eval function.」の1文が見られました。つまり、JavaScript1.5では、再度ビルトイン関数になったのです。

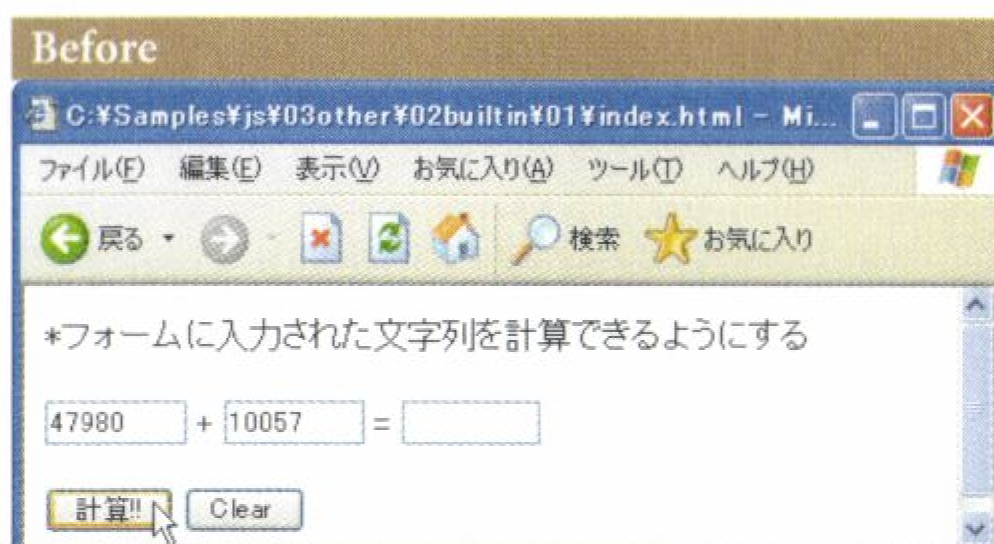
ちなみに、「eval()」はメソッドであっても、すべてのオブジェクトで使えるようになっていたので、実際はビルトイン関数と同じ感覚で使うことができました。しかし、すべてはECMAScriptと仕様を統一するためとはいえ、「ちょっとこれは変わりすぎではないか」という印象は否めません。今は、紆余曲折の末に元に戻った状態なので、このまま仕様が落ち着いてくれれば本当にいいのですが。

JavaScript

# フォームに入力された文字列を計算できるようにする

## eval() ［ビルトイン関数］

Before




After




ビルトイン関数「eval()」は、設定した文字列をJavaScriptのコードとてして評価します。フォームに入力された値は文字列ですので、そのままでは計算することができません。このような場合は、「eval()」を使って、文字列の内容を数値や演算子として評価し、JavaScriptで計算できるようにしてから計算する必要があります。

サンプルでは、フォーム「X」と「Y」に入力された文字列を、「eval()」を使って数値として評価した後で、その数値を加算して、結果をフォーム「Z」に表示しています。入力された値を計算するスクリプトは、JavaScriptの中でもよく利用されるものです。フォームの値をJavaScriptで計算する時は、「eval()」を使うことを忘れないようにしてください。

「eval()」は、たとえば「eval("4+4")」と文字列を評価した場合は「4+4」を計算した「8」の値が返りますが、「eval(new String("4+4"))」とStringオブジェクトを評価した場合は、文字列のまま「4+4」の値が返ります。さらに、文字列で表された演算子や、それにともなう一連の計算式のほか、変数やプロパティの評価も可能です。

「eval()」の扱いについては、コラム「「eval()」の変遷」(左ページ)を参照してください。

## Sample

```
<script type="text/javascript">
function calc(CL) { CL.Z.value = eval(CL.X.value)+eval(CL.Y.value) }
</script>
  ～中略～
<p>
<form name="Calc">
<input type="text" name="X" value="0" size="10">
+
<input type="text" name="Y" value="0" size="10">
=
<input type="text" name="Z" size="10">
</p>
<p>
<input type="button" value=" 計算!! " onclick="calc(this.form)">
<input type="reset" value=" Clear ">
</form>
</p>
```

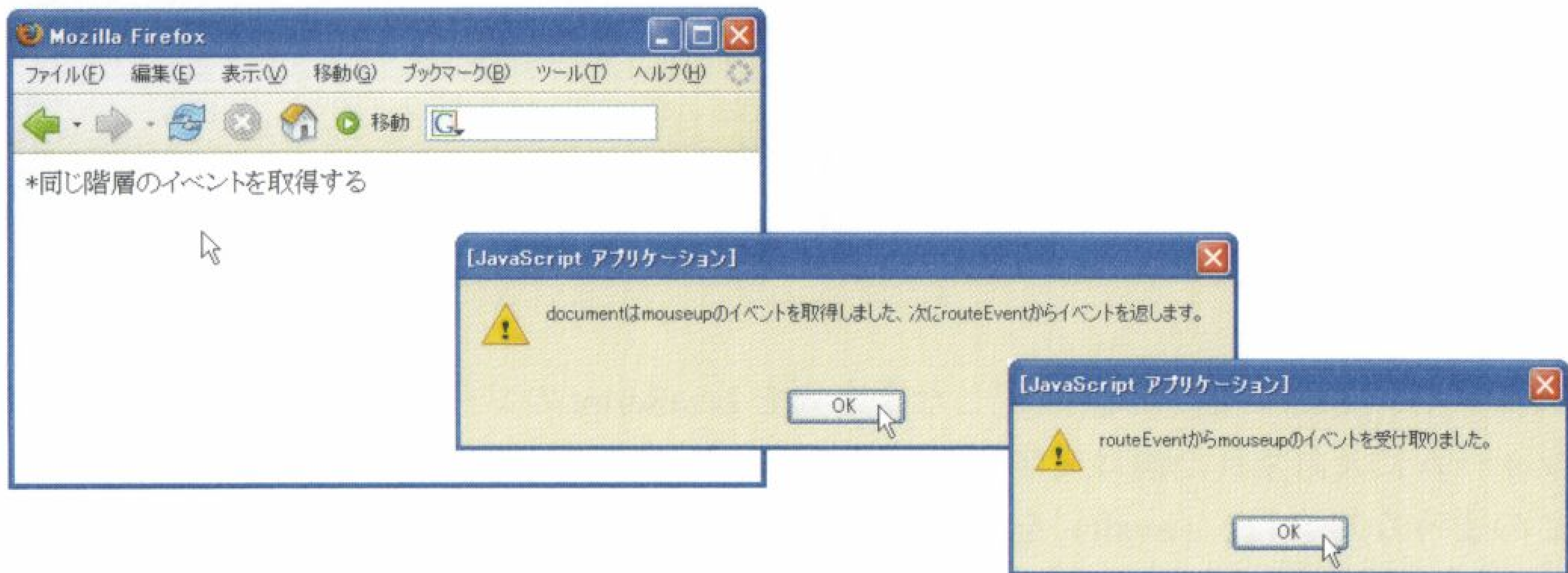
JavaScript

# 同じ階層のイベントを取得する

**オブジェクト名.routeEvent(Event.イベントタイプ)** [メソッド]

【サポートしているオブジェクト】
Window, document, layer

Firefox
Mozilla
N7.X
N6.X
N4.06
N4.X




「routeEvent()」メソッドは、「captureEvents()」メソッドで取得したのと同じイベントを返します。
サンプルでは、「routeEvent(e)」でマウスボタンが離された時に、取得されたイベントと同等のイベントを返します。
JavaScript1.2で追加されたメソッドです。

## Sample

```
<script type="text/javascript">
<!--
function fun1(e) {
    alert ("documentは" + e.type + "のイベントを取得しました。次にroute
Eventからイベントを返します。");
    document.routeEvent(e);
    alert ("routeEventから"+ e.type+"のイベントを受け取りました。" );
    return true;
}
document.captureEvents(Event.onmouseup);
document.onmouseup=fun1;
//-->
</script>
```

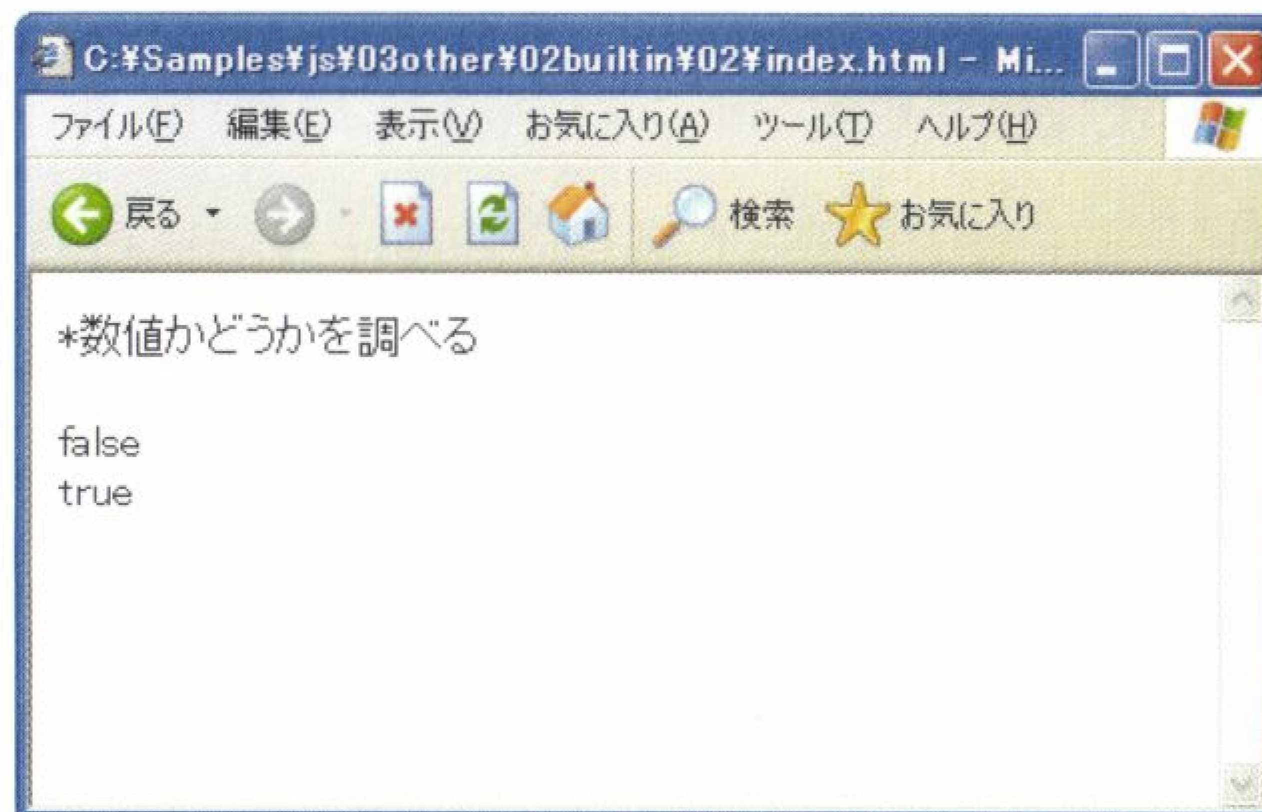
onmouseup:リファレンス「イベントタイプ」の「MouseUp」(P.624)

複数で利用する

# 数値かどうかを調べる

**isNaN()** [ビルトイン関数]






ビルトイン関数「isNaN()」は、与えられた値が数値かどうかを調べます。
値が数値の場合は「false」を、数値でない場合は「true」を返します。
バージョン3.X以前のNetscape Navigatorでは、UNIX版以外は機能しません。

## Sample

```
<script type="text/javascript">
<!--
var n = 1.23;
var m = "Mozi";
document.write(isNaN(n));
document.write("<br>");
document.write(isNaN(m));
//-->
</script>
```

# 有限数かどうかを調べる

## isFinite() [ビルトイン関数]




ビルトイン関数「isFinite()」は、与えられた値が有限数かどうかを調べます。
値が有限数の場合は「true」を、そうでない場合は「false」を返します。
サンプルでは、有限数である「1.23」と無限大を表す数値「Infinity」を、「isFinite()」を使って評価しています。「!m」は「Infinity」ではないということになるので、「true」の値が返ります。
JavaScript1.3で追加されたビルトイン関数です。

### Sample

```
<script type="text/javascript">
<!--
var n = 1.23;
var m = Infinity;
document.write(isFinite(n));
document.write("<br>");
document.write(isFinite(m));
document.write("<br>");
document.write(isFinite(!m));
//-->
</script>
```

# 文字列を整数に変換する

**parseInt()** [ビルトイン関数]




ビルトイン関数「parseInt()」は、文字列を整数に変換します。
小数点以下は切り捨てられ、数値でない場合は「NaN」を返します。
また、「parseInt()」は、「parseInt(文字列,n)」の用法を使うと、文字列をn進数として値を返します。たとえば、16進数で「parseInt("0xf",16)」または「parseInt("f",16)」、8進数で「parseInt("017",8)」または「parseInt("17",8)」、2進数で「parseInt("1111",2)」、10進数で「parseInt("15",10)」は、すべて「15」の値が返ります。

## Sample

```
<script type="text/javascript">
<!--
var n = "1.234";
var m = "0.777";
  document.write(parseInt(n));
  document.write("<br>");
  document.write(parseInt(m));
  document.write("<br>");
  document.write(parseInt(n)+parseInt(m));
  document.write("<br>");
  document.write(parseFloat("文字といってもこれだと..."));
//-->
</script>
```

IE6.0 IE5.5 IE5.0 IE4.0 Firefox Mozilla N7.X N6.X N4.06 N4.X Opera7 Opera6 Safari IE5-mac IE4-mac

# 文字列を浮動小数点数に変換する

**parseFloat()** [ビルトイン関数]




ビルトイン関数「parseFloat()」は、文字列を浮動小数点数に変換します。
前項の「parseInt()」とは違い、少数点以下は切り捨てられません。数値でない場合は「NaN」を返します。

## Sample

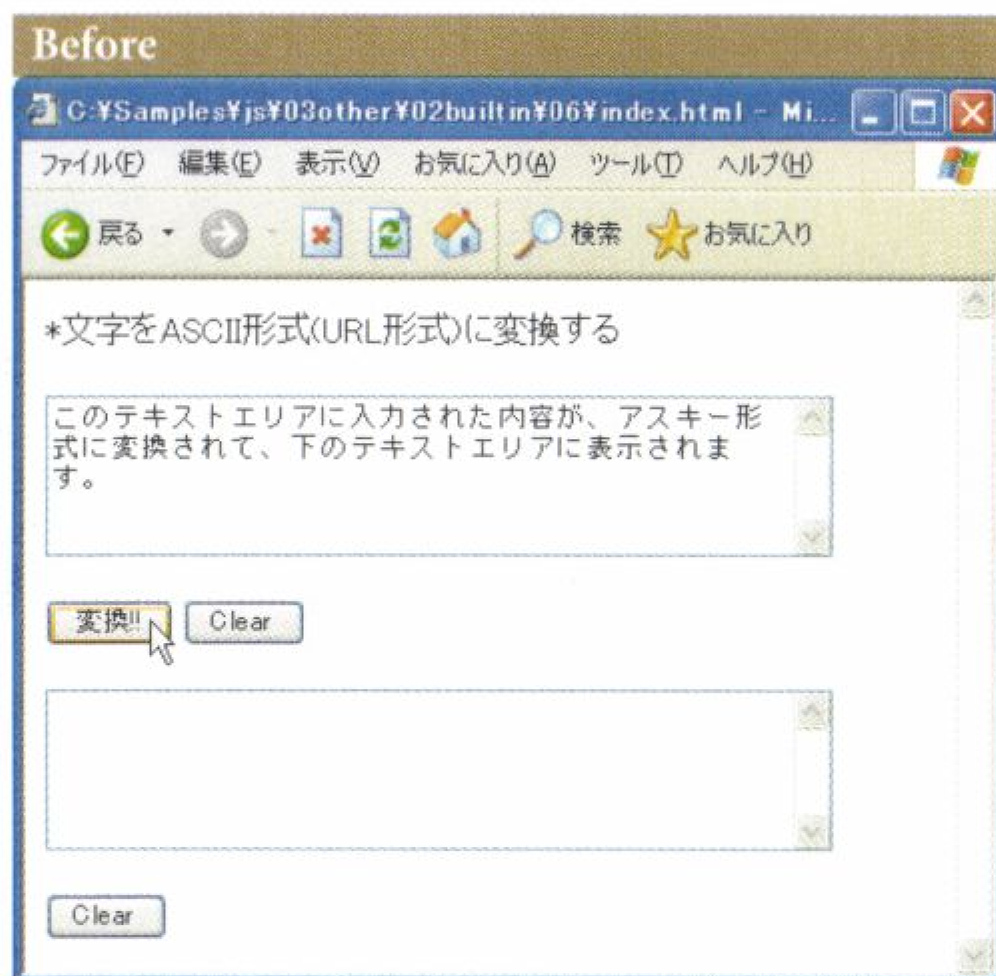
```
<script type="text/javascript">
<!--
var n = "1.234";
var m = "0.777";
  document.write(parseFloat(n));
  document.write("<br>");
  document.write(parseFloat(m));
  document.write("<br>");
  document.write(parseFloat(n)+parseFloat(m));
  document.write("<br>");
  document.write(parseFloat("文字といってもこれだと..."));
//-->
</script>
```

# 文字をASCII形式(URL形式)に変換する

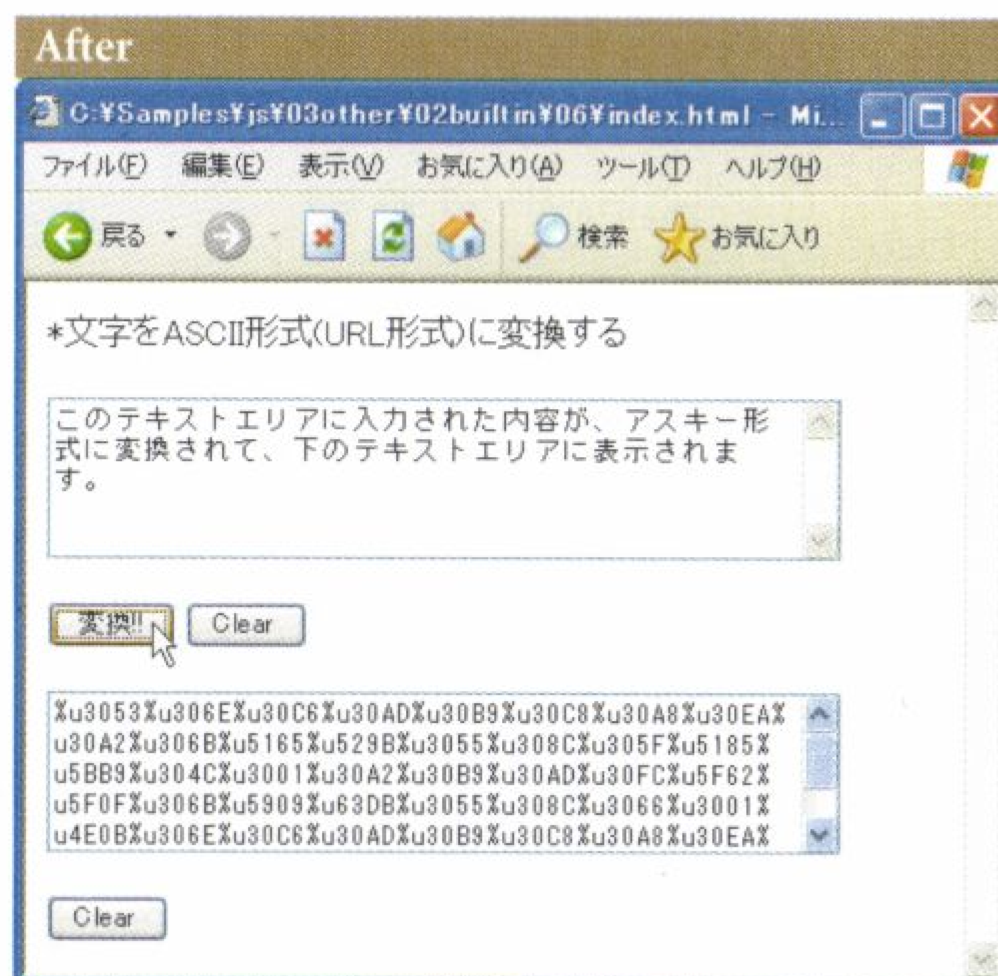
## escape()　[ビルトイン関数]

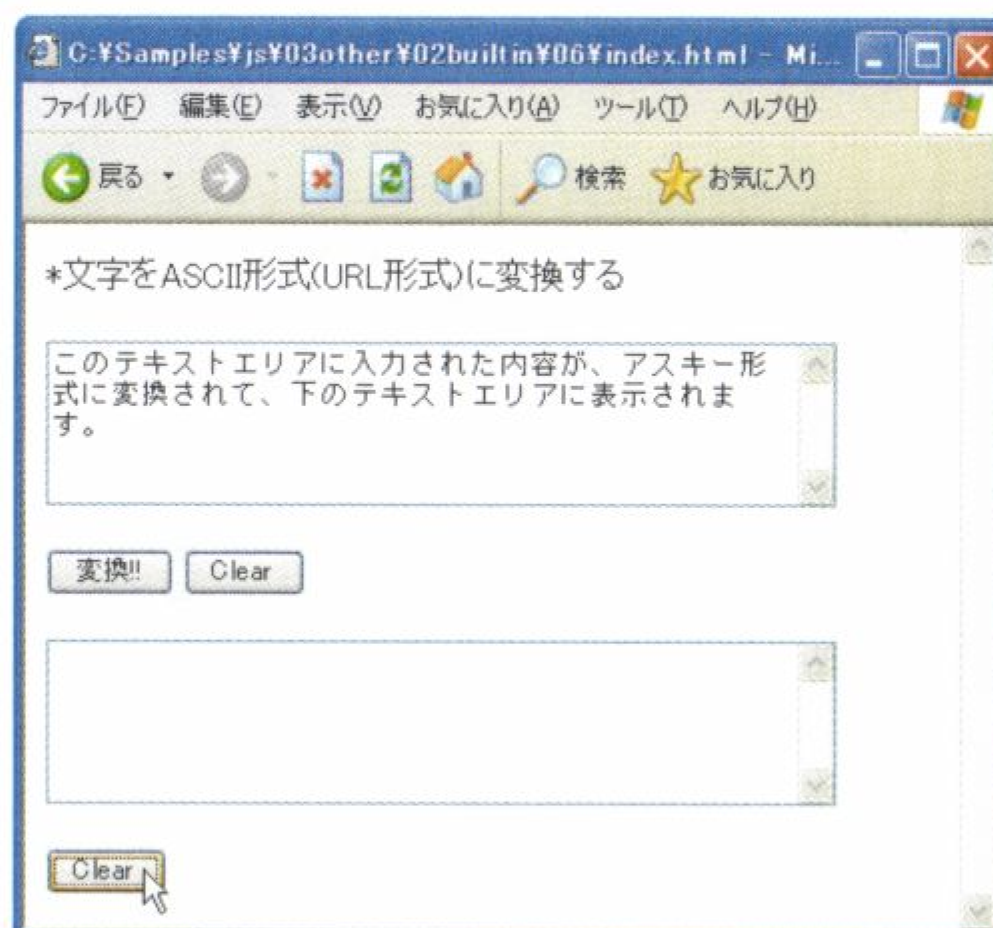


Before




After




ビルトイン関数「escape()」は、文字をASCII形式の文字に変換します。
サンプルでは、上のフォームに文字を入力して[変換!!]ボタンを押すと、下のフォームにASCII形式に変換された文字が表示されます。

## Sample

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
```

```
function dec1(C1) { document.Out1.Decode1.value = escape(C1.Input1
.value) }
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*文字をASCII形式(URL形式)に変換する
<form name="In1">
<textarea name="Input1" rows="5" cols="50">
</textarea>
<p>
<input type="button" value=" 変換!! " onclick="dec1(this.form)">
<input type="reset" value=" Clear ">
</p>
</form>
<form name="Out1">
<textarea name="Decode1" rows="5" cols="50">
</textarea>
<p>
<input type="reset" value=" Clear ">
</p>
</form>
</body></html>
```

**注意**

### それはJavaScriptのせいではないんだけど

よく聞かれる質問で、『フォームから送られるデータを「mailto:」オプションでメールで受け取るようにしているけど、メールで受けたフォームのデータが文字化けして読めない』というものがあります。フォーム内にJavaScriptを設定したサンプルをWebページ上で公開しているので、このような質問が来るのでしょうが、これに関してまず明確にしておくべきことは、「これはJavaScriptとは何にも関係がない」ということと、「その文字は別に文字化けしているわけではない」ということです。

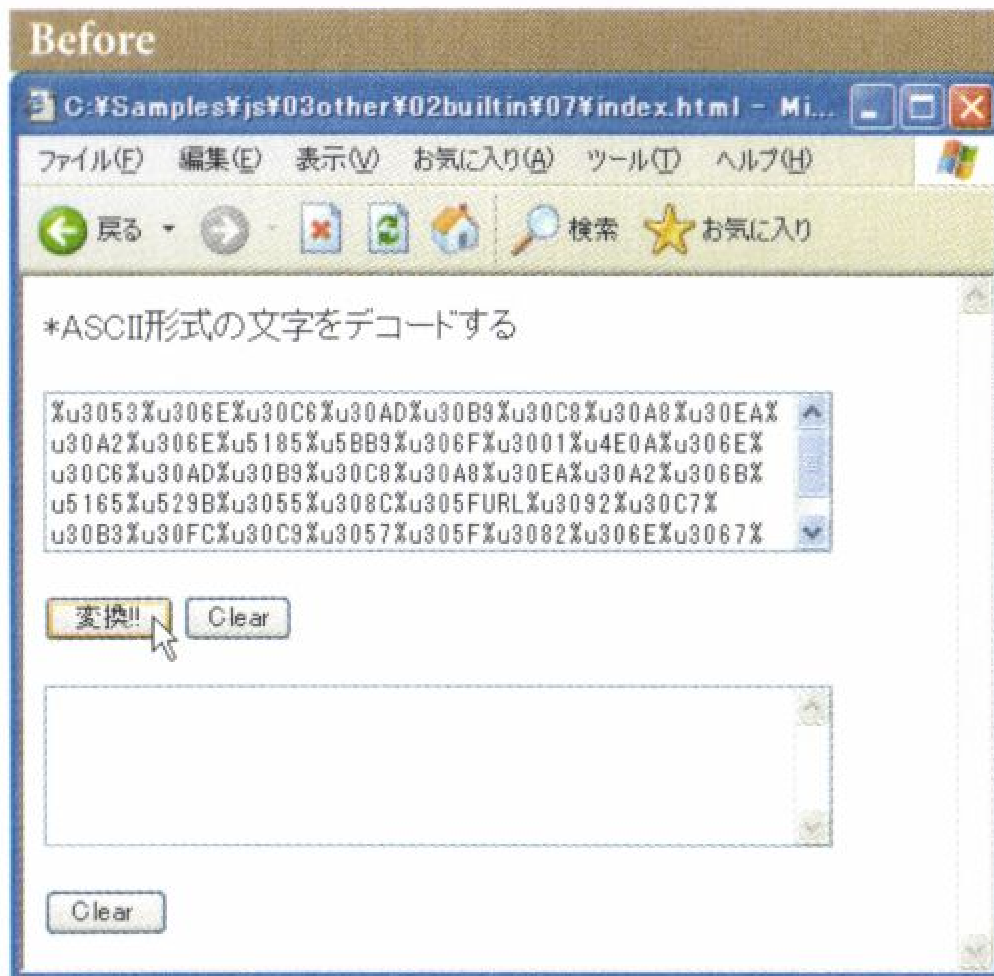
フォームから送られるデータは、URL形式という形式に変換して送られます。フォームから送られてくる「%」付きの文字が、これになります。したがって、受け取ったデータは、元の読めるような形式にデコードする必要があります。デコードするツールには、「ClipDecoder」などがあります。

# ASCII形式の文字をデコードする

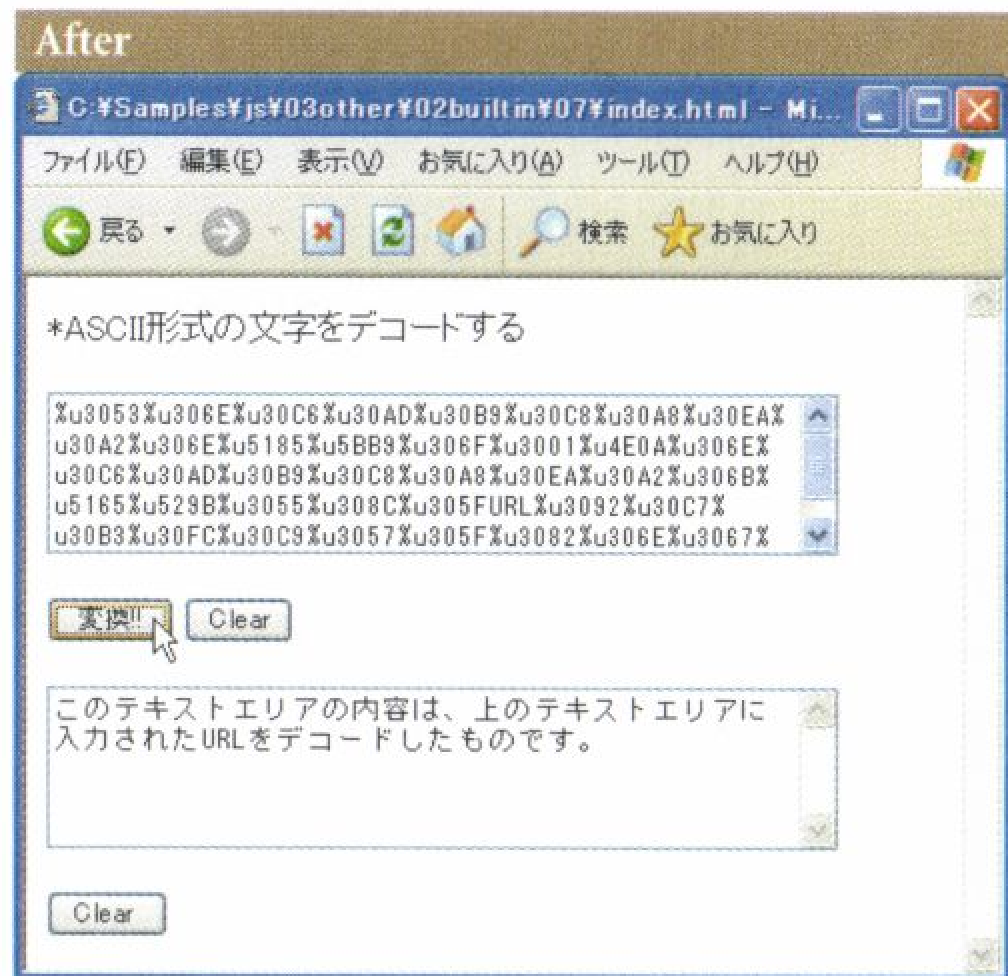
**unescape()** [ビルトイン関数]

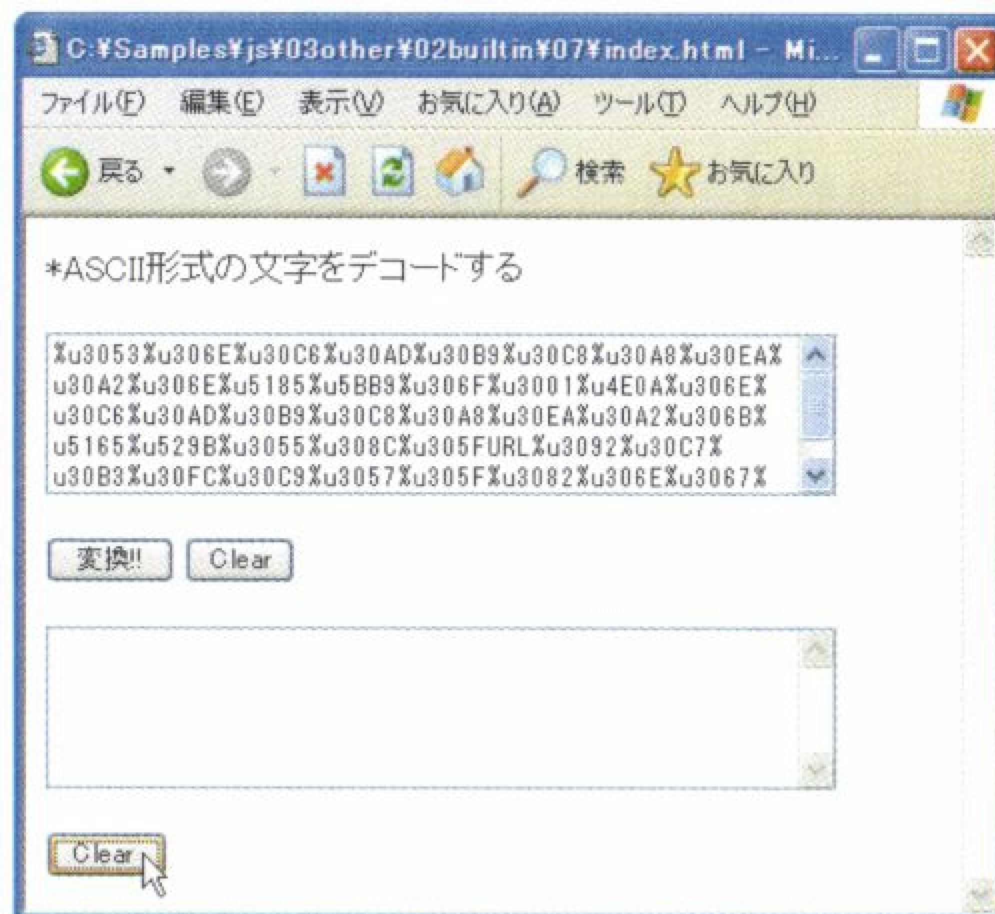


Before




After




ビルトイン関数「unescape()」は、ASCII形式の文字を文字に変換します。
サンプルでは、上のフォームにASCII形式の文字(フォームから送られてくる%付きの文字)を入力して[変換!!]ボタンを押すと、下のフォームに通常の文字に変換された文字が表示されます。

**Sample**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>

<script type="text/javascript">
<!--
function dec(C) { document.Out2.Decode2.value = unescape(C.Input
2.value) }
//-->
</script>

<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>

</head>
<body>
*ASCII形式の文字をデコードする
<form name="In2">
<textarea name="Input2" rows="5" cols="50">
</textarea>
<p>
<input type="button" value=" 変換!! " onclick="dec(this.form)">
<input type="reset" value=" Clear" >
</p>
<p></p>
</form>
<form name="Out2">
<textarea name="Decode2" rows="5" cols="50">
</textarea>
<p>
<input type="reset" value=" Clear ">
</p>
</form>
</body>
</html>
```

# その他のビルトイン関数

## taint()　[ビルトイン関数]

JavaScriptでは、セキュリティのため、通常は別のウィンドウやフレームからプロパティなどの値を参照できないようになっています。ビルトイン関数「taint()」は、一時的にその規制を無効にします。
参照可能にできるのは、プロパティや関数、オブジェクトなどの値です。また、「taint()」が有効な状態かどうかを調べるためには、navigatorオブジェクトの「taintEnabled()」メソッドを使用します。
「taint()」が有効な時のみ使用可能なプロパティとしては、documentオブジェクトの「domain」プロパティ、historyオブジェクトの「current」プロパティや「next」プロパティ、「previous」プロパティがあります。
Netscape Navigator 3.0で対応したJavaScript1.1で追加されたビルトイン関数ですが、Netscape Navigator 4.0からは、セキュリティ対策はSigned Scriptを使うことになったため、JavaScript1.2では削除されました。
通常のブラウザでは機能しません。

## untaint()　[ビルトイン関数]

ビルトイン関数「untaint()」は、「taint()」によって一時的に別のウィンドウやフレームから参照可能になったプロパティなどの値を、参照できない状態に戻します。
Netscape Navigator 3.0で対応したJavaScript1.1で追加されたビルトイン関数ですが、Netscape Navigator 4.0からは、セキュリティ対策はSigned Scriptを使うことになったため、JavaScript1.2では削除されました。
通常のブラウザでは機能しません。

# JavaScriptで取り扱える型の種類

JavaScript内で取り扱える値のタイプは、次の通りです。

## 文字列

ダブルクォーテーションマーク「"」、またはシングルクォーテーションマーク「'」で囲まれた文字や数字のことをいいます。

【例】 "123" , "Moziretu" , '文字列'

以下の特殊記号も文字列です。

| | |
|---|---|
| ¥b | バックスペース |
| ¥f | フォームフィード |
| ¥n | 改行 |
| ¥r | キャリッジリターン |
| ¥t | タブ |
| ¥¥ | バックスラッシュ |

(JavaScript1.5)

| | |
|---|---|
| ¥v | バーティカルタブ |
| ¥' | アポストロフィーまたはシングルクォーテーション |
| ¥XXX | 8進数で指定された、Latin-1の文字コードによって表された文字。XXXは、0から337までの3つまでの8進数。 |
| ¥xXX | 16進数で指定された、Latin-1の文字コードによって表された文字。XXは、00からFFまでの2つの16進数。 |
| ¥uXXXX | Unicodeの文字コードによって表された文字。XXXXは、4つの16進数。 |

(ユニコードエスケープシークェンス)

| | |
|---|---|
| ¥u0009 | タブ |
| ¥u000B | バーティカルタブ |
| ¥u000C | フォームフィード |
| ¥u0020 | スペース |
| ¥u000A | ラインフィード |
| ¥u000D | キャリッジリターン |
| ¥u0008 | バックスペース |
| ¥u0009 | ホリゾンタルタブ |
| ¥u0022 | ダブルクォーテーション |
| ¥u0027 | シングルクォーテーション |
| ¥u005C | バックスペース |

## 数値

**・整数**

8進数、10進数、16進数が使えます。

8進数は0から始まる0から7までで表す数値のことを、10進数は0以外から始まる0から9までで表す数値のことを、16進数は0xから始まる0から9までの数値と、それに続くaからf、あるいはAからFまでのアルファベットで表す数値のことをいいます。

【例】 0514 , 156 , 0x11

**・浮動少数点数**

小数点をピリオド(.)で表わした10進数、またはeあるいはEを使用する指数が使えます。

【例】 3.14 , 1.79E+308

## 論理値

値を比較した時、その比較が論理的に考えて正しいかどうかの値です。

正しければ「真」となり、正しくない場合は「偽」となります。たとえば「1==1」は正しいので「真」となり、「1==2」は正しくないので「偽」となります(「==」に関しては、右ページを参照してください)。

| | |
|---|---|
| true | 真の値 |
| false | 偽の値 |

## null値

プロパティなどに値が定義されていなかったり、何も設定されていない状態を表します。

| | |
|---|---|
| null | 未定義。何も設定されていない状態 |

# 演算子

JavaScriptで使用できる値の計算や、比較などに用いる記号は、次の通りです。

## 算術演算子

算術演算をするための演算子。

| | |
|---|---|
| = | 変数に値を代入する |
| + | 加算 |
| - | 減算または負の値を表す |
| * | 乗算 |
| / | 除算 |
| % | 乗除や除算で余りを求める |
| ++ | 値を1増やす(インクリメント) |
| -- | 値から1引く(デクリメント) |

インクリメントやデクリメントの用法は、次の通りです。

| | |
|---|---|
| y = x++ | yに値を代入してからxに1を加える |
| y = x-- | yに値を代入してからxから1引く |
| y = ++x | xに1加えてからyに値を代入する |
| y = --x | xから1引いてからyに値を代入する |

## 比較演算子

左辺と右辺の値を比較して、真の時は「true」の値を、偽の時は「false」の値を返す。

| | |
|---|---|
| x == y | xはyと等しい |
| x != y | xとyとは等しくない |
| x < y | xはyより小さい |
| x <= y | xはyより小さいか等しい |
| x > y | xはyより大きい |
| x >= y | xはyより大きいか等しい |

(JavaScript1.3)

| | |
|---|---|
| x === y | xはyと等しい。形の変更を行わない |
| x !== y | xとyとは等しくない。形の変更を行わない |

## 論理演算子

左辺と右辺の論理演算し、真の時は「true」の偽の時は「false」の値を返す。

| | |
|---|---|
| x && y | AND(xかつy) |
| x ¦¦ y | OR(xまたはy) |
| x !y | NOT(xはyでない) |

## 代入演算子

右辺の値を左辺に代入する演算子。

| | |
|---|---|
| x += y | 左辺に右辺の値を加算し結果を左辺に代入(x=x+yと同じ) |
| x -= y | 左辺に右辺の値を減算し結果を左辺に代入(x=x-yと同じ) |
| x *= y | 左辺に右辺の値を乗算し結果を左辺に代入(x=x*yと同じ) |
| x /= y | 左辺に右辺の値を除算し結果を左辺に代入(x=x/yと同じ) |
| x %= y | 左辺に右辺の値を除算し余りを左辺に代入(x=x%yと同じ) |

## 文字列演算子

文字列の連結を行う演算子。

| | |
|---|---|
| "文字列A" + "文字列B" | 「文字列A」と「文字列B」を連結する |
| a += "文字列B" | aの後に「文字列」を追加する |

## ビット演算子

値をビット単位で演算する。

コンピュータは、文字列も数値もすべてビット単位(2進数)で処理しています。ビット演算子は、値をコンピュータと同じように、ビット単位で取り扱う演算子です。

| | |
|---|---|
| ~ | ビットの反転 |
| & | ビットの論理積(AND) |
| ¦ | ビットの論理和(OR) |
| ^ | ビットの排他的和(XOR) |
| << | ビットの左シフト |
| >> | ビットの右シフト |
| >>> | ビットの論理右シフト |
| <<= | ビットごとの左シフトの代入 |
| >>= | ビットごとの右シフトの代入 |
| >>>= | 論理右シフトの代入 |

## カッコとその他

式や値を括るカッコや、オブジェクトやメソッドを区切るピリオドなども演算子です。

| | |
|---|---|
| [ ] | 配列の添番を括る |

| | |
|---|---|
| () | 値や数式を括る |
| . | オブジェクト、プロパティ、メソッドを区切る |
| , | コンマの両方の値や数式を評価する |

## new演算子

オブジェクトの作成(インスタンスの作成)を行う演算子。

```
オブジェクト名 = new オブジェクトタイプ(値1,値2,...値n)
```

【例】

```
<script language="JavaScript">
<!--
NewDay = new Date("may 2, 1997 23:00:00");
document.write(NewDay);
document.write("<br>");
NewDay.setDate(23);
document.write(NewDay);
//-->
</script>
```

## this演算子

this(キーワード)の後に指定したオブジェクトの参照を行う演算子。

オブジェクトは、継承関係を気にすることなく、直接指定できます。

```
this.オブジェクト
```

【例】

```
<script language="JavaScript">
<!--
function  Namae(n) { alert("入力された名前は"+n+"はです!!") }
//-->
</script>
名前を入力してください<br>
<form name="namae">
<input type="Text" name="namae" onBlur="Namae(this.value)" value="">
</form>
```

## 条件演算子

条件式が真(true)の時と偽(false)の時で、違った処理を行う演算子。

| | |
|---|---|
| 条件式 ? x:y | 条件式を評価して、真(true)の時はxの値を、偽(false)の時はyの値を参照する |

## typeof()演算子(JavaScript1.1)

数値、文字列、変数、オブジェクトなどの型(タイプ)を調べる演算子。

【例】

```
var suuzi = 123;
var mozi = "今日は";
var kye = null;
var today = new Date();
var kan = blur;
document.write(typeof(suuzi) + "<br>");
document.write(typeof(mozi) + "<br>");
document.write(typeof(kye) + "<br>");
document.write(typeof(today) + "<br>");
document.write(typeof(kan) + "<br>");
document.write(typeof(muimi));
```

## void()演算子(JavaScript1.1)

値を返さずに式を評価する演算子。

【例】

```
<a href="javascript:void(Kansuu())">この文字をクリックするとKansuu()が発生します</a>
```

## delete演算子(JavaScript1.2)

指定したオブジェクト、プロパティ、配列の要素の削除を行う演算子。

「delete」の処理が正常に行われた場合は「undefined」と「true」の値を返し、そうでなければ「false」の値を返します。

```
delete オブジェクト名
delete オブジェクト名.プロパティ名
delete 配列名[インデックス]
```

【例】では「delete」演算子を使って「hairetu[2]」の配列の要素を削除しているので、「document.write()」で書き出された結果は「0番目/1番目//3番目」となります。

**【例】**

```
<script language="JavaScript1.2">
<!--
hairetu= new Array("0番目","1番目","2番目","3番目");
delete hairetu[2];
document.write(hairetu.join("/"));
//-->
</script>
```

## in演算子(JavaScript1.4)

オブジェクト内に指定した値が含まれているかどうかを調べる演算子。
値が含まれている場合は、真(true)を返す。

```
プロパティ名または数値 in オブジェクト名
```

## instanceof演算子(JavaScript1.4)

オブジェクトのタイプを調べる演算子。
オブジェクトが指定したオブジェクトタイプの場合は、真(true)を返す。

```
オブジェクト名 instanceof オブジェクトタイプ
```

**【例】**

```
color1=new String("red");
color1 instanceof String; // trueが返る
color2="coral";
color2 instanceof String; // color はstringオブジェクトではないのでfalseが返る
```

## function演算子(JavaScript1.5)

関数の定義を行う演算子。

```
var 変数名 = function(引数) {処理内容};
```

## 演算子の優先順位

それぞれの演算子には優先順位があります。
上に行くほど優先順位が高くなります。

. []
() new
! ~ - + ++ -- typeof void delete
* / %
+ -
<< >> >>>
< <= > >= in instanceof
== != === !==
&
^
¦
&&
¦¦
?:
= += -= *= /= %= <<= >>= >>>= &= ^= ¦=
,

注意

## 文字列型と数値型の取り扱いについて

JavaScriptの文字列型と数値型の区別は、非常に曖昧です。

たとえば、次のスクリプト場合は、本当なら"2"は文字列型で、2は数値型なので「false」になるはずなのですが、JavaScript1.2以外では「true」の値を返します。

```
<script language="JavaScript">
<!--
document.write("2" == 2)
//-->
</script>
```

この時に注意が必要なのは、JavaScript1.2では、文字列型と数値型を明確に区別することです。

たとえば、次のスクリプト場合は「false」の値が返ります。

```
<script language="JavaScript1.2">
<!--
document.write("2" == 2)
//-->
</script>
```

しかし、JavaScript1.2でも次の例のように文字列に-0を設定すると、その他のバージョンのJavaScriptと同じように「true」の値を返すようになります。

```
<script language="JavaScript1.2">
<!--
document.write(("2"-0) == 2)
//-->
</script>
```

このように、JavaScript1.2で明確に区別されるようになった文字列型と数値型なのですが、JavaScript1.3以降になって、再びそれ以前のバージョンのJavaScriptのように区別されなくなりました。

たとえば、次のスクリプト場合は、JavaScript1.2以前のバージョンと同様に「true」の値が返ります。

```
<script language="JavaScript1.3">
<!--
document.write("2" == 2)
//-->
</script>
```

これは、ECMAScriptと仕様を合わせるためです。

JavaScript1.3以降でJavaScript1.2のように文字列型と数値型を区別して値を比較したい時には、JavaScript1.3で新たに追加された演算子「===」や「!==」を使用します。

たとえば、次のように値の比較に「===」を使用すると、JavaScript1.2のように文字列型と数値型を明確に区別して「false」の値が返ります。

```
<script language="JavaScript1.3">
<!--
document.write("2" === 2)
//-->
</script>
```

# JavaScriptの命令文(ステートメント)

JavaScriptで使用できる命令文は、次の通りです。

## コメント

JavaScript内にコメントを書く時に使用します。
「//」はそれ以降の1行が、「/* ~ */」は間に挟まれた文字列が、コメントとして扱われます。

```
//
/*   */
```

【例】

```
//この1行はコメントとなります。

/*これに囲まれている部分は
  コメントとなります。*/
```

## ループから抜ける「break」

「for」や「while」などで、繰り返し(ループ)処理を行っている時に使用します。
「条件式」が真(true)の時は、1番内側のループを抜けます。

```
brerk;
```

【例】

```
for (i=1; i<=10; i++) {
    document.write("この文章を10回書き出します:" +i+ "回目");
        if (i==5);
        break;
    document.write("でも...。<br>");
}
document.write("<br>breakがあるので5回でループを抜けます<br>");
```

## 処理のスキップ「continue」

「for」や「while」などで、繰り返し(ループ)処理を行っている時に使用します。
「条件式」が真(true)の時、「continue」以下の処理を飛ばして、繰り返し処理を続けます。

```
continue;
```

【例】

```
for (i=1; i<=10; i++) {
        if (i==5);
        continue;
    document.write("この文章を10回書き出します:" +i+ "回目でも...。<br>");
}
document.write("<br>continueがあるので5回目がありません<br>");
```

## 繰り返し処理「for」

「条件式」が真(true)の間、「処理」を繰り返し行います。

```
for (初期値; 条件式; 増減式 ) { 処理 }
```

【例】

```
for (i=1; i<=4; i++) {
    document.write("この文章を4回書き出します:" +i+ "回目<br>");
}
```

**重要**

### コメントを使用する時の注意

JavaScript内で「<!--HTMLのコメント-->」を使っても、エラーにはなりません。
しかし、JavaScriptを無効にしている、あるいは未対応のブラウザで見ると、「<!--HTMLのコメント-->」以下のソースコードが、隠されずに丸見えになってしまいます。

## プロパティ、メソッドの一覧「for...in」

「オブジェクト」内のプロパティ・メソッドを「変数」に代入しながら順番に取り出します。

```
for (変数 in オブジェクト) { 処理 }
```

【例】

```
for (var n in navigator){
    document.write(n,"<br>");
}
```

## 関数の設定「function」

JavaScriptでは、一定の処理手続きを「関数」として設定することができます。

「function」の後に「関数名」とその「処理」を記述し、ページが読み込まれたタイミングや、イベントハンドラによってイベントが発生したタイミングで、「関数名」を呼ぶことによって「処理」が実行されます。

通常、関数の設定は、HTMLファイルの<head>～</head>内で行います。これは、関数が設定される前に、<body>～</body>内で設定する「関数名」が呼ばれるのを防ぐためです。

```
function 関数名([引数] [, 引数] [...,
引数]) { 処理 }
```

【例】

```
function kansuu(x,y) {
    document.write( x +"足す" + y +
"は" + eval(x+y) +"です");
}
kansuu(1,2);
```

## 条件分岐「if」

「条件式」が真(true)の時は「処理1」を行い、それ以外の時は「処理2」を行います。

「else」の処理がない時は省略可能です。

```
if (条件式) { 処理1 }
    else{ 処理2 }
```

【例】

```
var now = new Date();
var AMPM = now.getHours();
    if (AMPM < 12 ){ document.write
("午前") }
        else { document.write("午後"
) }
```

## 値の代入「var」

変数、配列、オブジェクトなどの宣言を行います。「=値」を設定すれば、「var名」にその値が代入されます。

```
var var名 [= 値] [..., var名 [= 値] ]
```

【例】

```
var ataiA =5;
var ataiB =3;
var ataiC ="足す";
document.write( ataiA + ataiC + ata
iB + "は" + eval(ataiA+ataiB) +"です
");
```

## 繰り返し処理「while」

「条件式」が真(true)の間、「処理」を繰り返し行います。

```
while( 条件式 ) { 処理 }
```

【例】

```
var i = 1
while ( i <= 4 ) {
    document.write("この文章を4回書き出
します:" +i+ "回目<br>");
    i++;
}
```

## オブジェクトの省略「with」

「オブジェクト」で指定したオブジェクトを省略して、プロパティやメソッドを使用できるようにします。

```
with (オブジェクト) { 処理 }
```

【例】

```
with (document) {
    write("この文章は、documentの<br>");
```

```
  write("部分を省略しています。<br>");
}
```

## 戻り値を返す「return」

命令文内で値を返す時に使用します。
返す値がない時は不要です。

```
return 値
```

**【例】**

```
<a href="../../home.html" onMouseOver="window.status='ステータス行にメッセージが出ます';return true">この文字の上にマウスカーソルを持ってくると</a>
```

## 配列の作成

「function MakeArray(n) {」から「return this ;}」までの処理で配列の要素を入れる器を作り、「要素名= new MakeArray(要素数);」以下で配列の要素を流し込み、配列を作成します。
作成された配列は、「要素名[添番]」で希望の要素を取り出すことができます。
Netscape Navigator 3.0で対応したJavaScript1.1からは、配列を扱うArrayオブジェクトが追加され、もっと簡単に配列を作成できるようになりました。また、Arrayオブジェクトの一部の機能は、Netscape Navigator 2.0でも使用可能です。詳しくは、「Arrayオブジェクト」(P.562～)を参考にしてください。
【例】は、「Dateオブジェクト」の「曜日を表示する」(P.502)のサンプルを、配列を使って書き直したものです。

```
function MakeArray(n) {
  this.length = n ;
  for (var i = 0; i <= n; i++) {
    this[i] = 0;
  }
  return this ;
}
要素名 = new MakeArray(要素数) ;
要素名[添番] = 要素 ;
  ⟆
```

**【例】**

```
<html>
<head>
<title>ARRAY_SAMPLE_1</title>
<script type="text/javascript">
<!--
function MakeArray(n) {
  this.length = n ;
  for (var i = 0; i <= n; i++) {
    this[i] = 0;
  }
  return this ;
}
youbi = new MakeArray(7) ;
youbi[0] = "日";
youbi[1] = "月";
youbi[2] = "火";
youbi[3] = "水";
youbi[4] = "木";
youbi[5] = "金";
youbi[6] = "土";
function gety(y){ document.write(youbi[y] ) }
//-->
</script>
</head>
<body>
*配列を使った曜日表示<p>
(
<script type="text/javascript">
<!--
day = new Date();
gety(day.getDay());
//-->
</script>
)
</body>
</html>
```

## HTMLを配列として扱う

リンクやフォームなどのHTMLのタグは、JavaScriptのオブジェクトとして取り扱えるのと同時に、配列の要素としても取り扱えます。

リファレンス

HTMLファイルに記述されている、それぞれのオブジェクトの上から、添番が[0][1][2]...となる配列の要素が作成されます。それらの要素は、「オブジェクトの配列名[添番]」で取り扱うことができます。

```
オブジェクトの配列名[添番]
```

**【オブジェクトの配列名】**

anchors, applets, elements, embeds, forms, frames, history, images, layers, links, mineTypes, options, plugins

※各配列には、配列の数を持ったlengthプロパティがあります。

**【例】**

```
<html>
<head>
<title>ARRAY_SAMPLE_2</title>
<script type="text/javascript">
<!--
function arr() {
    document.forms[0].elements[0].v
alue= "formのelements配列[0]の要素";
    document.forms[0].elements[1].v
alue= "formのelements配列[1]の要素";
    document.forms[0].elements[2].v
alue = "formのelements配列[2]の要素";
    document.forms[0].elements[3].v
alue = "formのelements配列[3]の要素";
}
//-->
</script>
</head>
<body onLoad="arr()">
*HTMLを配列として捉える<p>
<form name="ARRAY" >
<input type="text" name="array1" si
ze=40>
<br>
<input type="text" name="array1" si
ze=40>
<br>
<input type="text" name="array1" si
ze=40>
<br>
<input type="text" name="array1" si
ze=40>
</form>
</body>
</html>
```

## ラベル(JavaScript1.2～)

式にラベルを設定しておくことにより、if文やfor文、while文のようなループ文から「break」や「continue」で抜ける時、ラベル名を指定して明示的に抜ける場所を設定することができます。

【例】では、まず変数「i」に「4」の値を、変数「j」に「2」の値を代入しています。そして、ラベル名で「checkiandj」と名前を付けたif文の処理と、ラベル名で「checkj」と名前を付けたif文の処理を用意しています。変数「k」の値が「1」の時は、条件式「k==1」が真(true)となり、「break checkj」で「checkj」の処理を抜けるため、その下の「i」と変数「j」の合計を出す処理を行なわずに「i-j」が実行されます。「k」の値が「1」以外の時は、「i」と「j」の合計を出した後に「break checkiandj」で「checkiandj」の処理を抜けるため、「i-j」の処理は実行されません。

```
ラベル名 :
     式
break ラベル名
continue ラベル名
```

**【例】**

```
var i=4;
var j=2;
checkiandj :
if (4==i) {
    document.write("kの値は" + k + "
.<br>");
        document.write("iの値は" + i
 + ".<br>");
        checkj :
            if (2==j) {
                document.write("jの
値は" + j + ".<br>");
                    if (1==k) { bre
ak checkj }
```

```
                document.write("iと
jの合計" + (i+j) + ".<br>");
                break checkiandj;
        }
        document.write(i + "-" + j
+ "=" + (i-j) + ".<br>");
    }
```

## 繰り返し処理「do while」(JavaScript1.2～)

do while文は、条件式が偽(false)になるまで、処理を繰り返し行なうような場合に使用します。

【例】では、「i<10」が偽(false)になるまで、iに1を加えながら書き出しています。

```
do 処理
    while (条件式);
```

**【例】**

```
var i=0;
do {
    i+=1;
    document.write(i,"<br>");
    } while (i<10);
```

## スイッチ「switch」(JavaScript1.2～)

switch文は、処理にラベルを設定し、各ラベルと値を参照して真(true)になる処理を実行します。この時、もしも真(true)になるラベルがない場合は、「default:」の処理を実行します。

【例】では、変数「i」の値が「Oranges」の時は「オレンジは1個100円です。」という文字が、「Apples」の時は「りんごは1個150円です。」という文字が、documentオブジェクトの「write()」メソッドによって書き出されます。また、「i」の値が「Oranges」でも「Apples」でもない場合は、「default:」で設定した、13行目の「申し訳ありません。ただいま品切れです。」という文字を書き出されます。

```
switch (値) {
    case ラベルA :
        処理;
        break;
    case ラベルB :
        処理;
        break;
          ∫
    default:
        処理;
}
```

**【例】**

```
switch (i) {
    case "Oranges" :
        document.write("オレンジは1個
100円です。<br>");
        break;
    case "Apples" :
        document.write("りんごは1個15
0円です。<br>");
        break;
    default:
        document.write("申し訳ありませ
ん。ただいま品切れです。");
}
document.write("なお、上記金額に消費税は含
まれておりません。<br>");
```

## エクスポート「export」(JavaScript1.2～)

Signed Scriptが、他のSigned Scriptあるいは署名なしのスクリプトに、関数・プロパティ・オブジェクトの使用を可能にします。

通常、Signed Script内の情報は、同じSigned Script内でしか使用することができません。exportは、このSigned Script内の情報を、他のスクリプトから利用可能にします。

```
export リスト名1, リスト名2, ..., リスト名n
export *
```

| | |
|---|---|
| リスト名 | エクスポートする関数·プロパティ·オブジェクト |
| * | Signed Script内のすべての関数·プロパティ·オブジェクト |

リファレンス

## インポート「import」（JavaScript1.2～）

スクリプトが、exportにより使用可能になった、関数・プロパティ・オブジェクトの使用を可能にします。

```
import オブジェクト名.リスト名1, オブジェクト名.リスト名2, ..., オブジェクト名.リスト名n
import オブジェクト名.*
```

| | |
|---|---|
| オブジェクト名 | 受け取るオブジェクト名 |
| リスト名 | 受け取る関数・プロパティ・オブジェクトのリスト |
| * | エクスポートされたすべての関数・プロパティ・オブジェクト |

## 例外の処理「try...catch」（JavaScript1.4～）

試みる処理のブロックをマークします。tryの処理から例外が投げられた時には、catchで設定した処理が行われます。

```
try {
    処理
}
[catch (例外の値) {tryから例外が投げられた時の処理}]...
[finally {tryの前に実行する処理、例外が投げられた時も実行する}]
```

**【例】**

```
try {
    throw "例外が発生したよ!!" //例外を生成します
}
catch (e) {
    //例外時の処理
    logMyErrors(e) //別の部分で設定したエラー処理へ
}
```

## 例外の値を投げる「throw」（JavaScript1.4～）

ユーザー定義の例外の値を投げます。

```
throw 例外の値
```

**【例】**

```
throw "Error2"; //文字列の値を持った例外を生成します
throw 42; //42の値を持った例外を生成します
throw true; //trueの値を持った例外を生成します
```

## 定数を宣言する「const」（JavaScript1.5～）

読み出し専用の定数を宣言します。

```
const 定数名 [= 値] [..., var名 [= 値] ]
```

**【例】**

```
const a = 10;
document.write( "定数10は"+a);
```

リファレンス

# イベントハンドラ

ユーザーやスクリプトによってページがロードされたり、オブジェクトがクリックされたりというような、特定の動作が起こったタイミングをイベントといいます。JavaScriptでは、イベントの発生を取得し、そのタイミングでスクリプトの実行を開始することができます。このイベントの取得を行う指定を、イベントハンドラといいます。

イベントハンドラの設定は、そのイベントハンドラが設定可能なオブジェクトのHTMLタグ内に設定することによって行います。

JavaScriptで用意されているイベントハンドラと、そのイベントハンドラがどのようなイベントを取得し、どのオブジェクトに対応しているかは、次の通りです。

## onAbort(JavaScript1.1～)

画像の読み込みがキャンセルされた時のイベントを取得するイベントハンドラ。

画像が読み込まれている途中に、ブラウザのストップボタンを押すなどの動作で、画像の読み込みがキャンセルされた時をイベントとして取得し、設定した処理を実行します。

JavaScript1.1からのイベントハンドラです。

**【利用可能なオブジェクト】**

(JavaScript1.1)

Imageオブジェクト

## onBlur(JavaScript1.0～)

フォーカスがフォームやウィンドウから離れた時のイベントを取得するイベントハンドラ。

Textフォームなどでマウスカーソルが点滅していたり、Radioフォームや Checkboxフォームがチェックされていたり、ウィンドウがアクティブになっているなどの状態が、フォーカスがある状態になります。

フォームを移動したり、フォーカスがあるウィンドウとは別ウィンドウをクリックすることなどで、フォーカスが移動した時をイベントとして取得し、設定した処理を実行します。

**【利用可能なオブジェクト】**

(JavaScript1.0)

Selectオブジェクト

Textオブジェクト

Textareaオブジェクト

(JavaScript1.1)

Buttonオブジェクト

Checkboxオブジェクト

FileUploadオブジェクト

Frameオブジェクト

Passwordオブジェクト

Radioオブジェクト

Resetオブジェクト

Submitオブジェクト

windowオブジェクト

## onChange(JavaScript1.0～)

フォームの内容に変化があり、フォーカスがフォームから離れた時のイベントを取得するイベントハンドラ。

Textフォームの内容を変更し、他のフォームに移動することなどで、フォーカスが移動した時をイベントとして取得し、設定した処理を実行します。

**【利用可能なオブジェクト】**

(JavaScript1.0)

Selectオブジェクト

Textオブジェクト

Textareaオブジェクト

(JavaScript1.1)

FileUploadオブジェクト

## onClick(JavaScript1.0～)

ボタンやリンクをクリックした時のイベントを取得するイベントハンドラ。

Buttonフォームやリンクをクリックした時をイベントとして取得し、設定した処理を実行します。

JavaScript1.1から「return false」と「false」を返すと、リンクの参照などの通常動作を中止できるようになりました。これにより、リンクにこのイベントハンドラを設定して、JavaScriptの処理の1番最後に「return false」を返すように設定することによって、リンクをクリックしてもJavaScriptのみを実行し、別ページに移動するなどのリンクの処理を行わないようにすることができます。

**【利用可能なオブジェクト】**

（JavaScript1.0）
Buttonオブジェクト
Checkboxオブジェクト
Linkオブジェクト
Radioオブジェクト
Resetオブジェクト
Submitオブジェクト

## onError（JavaScript1.1～）

ページや画像の読み込みエラーが発生した時のイベントを取得するイベントハンドラ。
画像読み込み時に、リンク切れなどで画像の読み込みがうまく行われなかった時をイベントとして取得し、設定した処理を実行します。
JavaScript1.1からのイベントハンドラです。

**【利用可能なオブジェクト】**

（JavaScript1.1）
Imageオブジェクト
windowオブジェクト

## onFocus（JavaScript1.0～）

フォーカスがフォームやウィンドウに与えられた時のイベントを取得するイベントハンドラ。
フォームを移動したり、フォーカスがないウィンドウをクリックするなどでフォーカスが移動した時をイベントとして取得し、設定した処理を実行します。

**【利用可能なオブジェクト】**

（JavaScript1.0）
Selectオブジェクト
Textオブジェクト
Textareaオブジェクト

（JavaScript1.1）
Buttonオブジェクト
Checkboxオブジェクト
FileUploadオブジェクト
Frameオブジェクト
Passwordオブジェクト
Radioオブジェクト
Resetオブジェクト
Submitオブジェクト
windowオブジェクト

## onLoad（JavaScript1.0～）

ページや画像が読み込まれた時のイベントを取得するイベントハンドラ。
ページや画像の読み込みが終了した時をイベントとして取得し、設定した処理を実行します。

**【利用可能なオブジェクト】**

（JavaScript1.0）
windowオブジェクト

（JavaScript1.1）
Imageオブジェクト

## onMouseOut（JavaScript1.1～）

マウスが指定された領域から離れた時のイベントを取得するイベントハンドラ。
リンクに設定することによって、マウスカーソルがリンクから離れた時をイベントとして取得し、設定した処理を実行します。
JavaScript1.1からのイベントハンドラです。

**【利用可能なオブジェクト】**

（JavaScript1.1）
Linkオブジェクト
Areaオブジェクト

## onMouseOver（JavaScript1.0～）

マウスが指定された領域内に入った時のイベントを取得するイベントハンドラ。
リンクに設定することによって、マウスカーソルがリンク上に乗った時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】
（JavaScript1.0）
Linkオブジェクト

（JavaScript1.1）
Areaオブジェクト

## onReset（JavaScript1.1～）

フォームがリセットされた時のイベントを取得するイベントハンドラ。
Resetフォームが押されるなどで、フォームの内容がリセットされた時をイベントとして取得し、設定した処理を実行します。
JavaScript1.1からのイベントハンドラです。

【利用可能なオブジェクト】
（JavaScript1.1）
Formオブジェクト

## onSelect（JavaScript1.0～）

フォームのテキスト領域が選択された時のイベントを取得するイベントハンドラ。
Textフォームや Textareaフォームが選択され、フォームへの入力が可能になった時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】
（JavaScript1.0）
Textオブジェクト
Textareaオブジェクト

## onSubmit（JavaScript1.0～）

フォームのSubmitボタンが押された時のイベントを取得するイベントハンドラ。
Submitボタンが押され、フォームのデータ送信が開始される時をイベントとして取得し、設定した処理を実行します。
データ送信の処理はJavaScriptの処理が終了するまで行われず、JavaScriptの処理で「return false」と「false」を返すと、データ送信の処理は中止されます。

【利用可能なオブジェクト】
（JavaScript1.0）
Formオブジェクト

## onUnload（JavaScript1.0～）

別のページに移動した時のイベントを取得するイベントハンドラ。
今のページを抜けて、別ページに移動した時をイベントとして取得し、設定した処理を実行します。

【利用可能なオブジェクト】
（JavaScript1.0）
windowオブジェクト

# イベントタイプ

JavaScript1.2からは、イベントをオブジェクトとして捕らえるeventオブジェクトが追加されました。そのため、イベントを取得したいオブジェクトに対してイベントを設定することによって、そのオブジェクト上のどこででもイベントの発生を取得することができるようになりました。

JavaScriptで設定できるイベントと、そのイベントが設定できるオブジェクト、イベントで取得できる値、つまりeventオブジェクトのプロパティは、次の通りです。

## Click

マウスがクリックされた時のイベントを取得します。「MouseDown」イベントと「MouseUp」イベントを合わせたものです。

```
onclick=関数名、またはスクリプト
```

**【利用可能なオブジェクト】**

Button, Checkbox, Link, Radio, Reset, Submit

**【eventオブジェクトのプロパティ】**

| | |
|---|---|
| type | 「click」を値に持つ |
| layerX, layerY, pageX, pageY, screenX, screenY | イベントが発生した時のカーソルのX軸Y軸位置を値に持つ |
| which | 左ボタンの時は「1」を、右ボタンの時は「3」を値に持つ |
| modifiers | 修飾キーの値を持つ |

※buttonがクリックされてもlayerX, layerY, pageX, pageY, screenX, screenYは値を返しません。

## DblClick

マウスがダブルクリックされた時のイベントを取得します。

WindowsとMacintoshで使用可能です。

```
ondblclick=関数名、またはスクリプト
```

**【利用可能なオブジェクト】**

document, Area, Link

**【eventオブジェクトのプロパティ】**

| | |
|---|---|
| type | 「dblclick」を値に持つ |
| layerX, layerY, pageX, pageY, screenX, screenY | イベントが発生した時のカーソルのX軸Y軸位置を値に持つ |
| which | 左ボタンの時は「1」を、右ボタンの時は「3」を値に持つ |
| modifiers | 修飾キーの値を持つ |

## DragDrop

ウィンドウ上にファイルやショートカットなどをドラッグ&ドロップした時のイベントを取得します。イベントが発生した時に、真(true)を返せばドラッグ&ドロップを許し、偽(false)を返せばドラッグ&ドロップの動作を中止します。

Windows版のNetscape Navigator 4.Xは、画像のドラッグ&ドロップに対応していません。

```
ondragdrop=関数名、またはスクリプト
```

**【利用可能なオブジェクト】**

window

**【eventオブジェクトのプロパティ】**

| | |
|---|---|
| type | 「dragdrop」を値に持つ |
| data | ドロップされたファイルなどのURLを返す |

## KeyDown

ユーザーがキーを押した時のイベントを取得します。

「KeyDown」イベントは「KeyPress」イベントより前に発生し、もし「KeyDown」イベントが偽(false)を返した時は、「KeyPress」イベントは発生しません。

```
onkeydown=関数名、またはスクリプト
```

**【利用可能なオブジェクト】**

document, Image, Link, Textarea

リファレンス

【eventオブジェクトのプロパティ】

| | |
|---|---|
| type | 「keydown」を値に持つ |
| layerX, layerY, pageX, pageY, screenX, screenY | イベントが発生した時のカーソルのX軸Y軸位置を値に持つ |
| which | 押されたキーのASCIIの値を持つ |
| modifiers | 修飾キーの値を持つ |

## KeyPress

ユーザーがキーを押したままの状態にした時のイベントを取得します。

「KeyDown」イベントが真(true)の値を返した時のみイベントが発生し、ユーザーがキーを放すまでイベントは発生し続けます。

```
onkeypress=関数名、またはスクリプト
```

【利用可能なオブジェクト】

document, Image, Link, Textarea

【eventオブジェクトのプロパティ】

| | |
|---|---|
| type | 「keypress」を値に持つ |
| layerX, layerY, pageX, pageY, screenX, screenY | イベントが発生した時のカーソルのX軸Y軸位置を値に持つ |
| which | 押されたキーのASCIIの値を持つ |
| modifiers | 修飾キーの値を持つ |

## KeyUp

ユーザーがキーを放した時のイベントを取得します。

```
onkeyup=関数名、またはスクリプト
```

【利用可能なオブジェクト】

document, Image, Link, Textarea

【eventオブジェクトのプロパティ】

| | |
|---|---|
| type | 「keyup」を値に持つ |
| layerX, layerY, pageX, pageY, screenX, screenY | イベントが発生した時のカーソルのX軸Y軸位置を値に持つ |
| which | 放されたキーのASCIIの値を持つ |
| modifiers | 修飾キーの値を持つ |

## MouseDown

ユーザーがマウスボタンを押した時のイベントを取得します。

もしも「MouseDown」イベントが偽(false)を返した時には、選択やリンクの参照などの通常動作が中止されます。

```
onmousedown=関数名、またはスクリプト
```

【利用可能なオブジェクト】

Button, document, Link

【eventオブジェクトのプロパティ】

| | |
|---|---|
| type | 「mousedown」を値に持つ |
| layerX, layerY, pageX, pageY, screenX, screenY | イベントが発生した時のカーソルのX軸Y軸位置を値に持つ |
| which | 左ボタンの時は「1」を、右ボタンの時は「3」を値に持つ |
| modifiers | イベントが発生した時に押された修飾キーの値を持つ |

## MouseMove

カーソルが動いた時のイベントを取得します。

「captureEvents()」メソッドでこのイベントを取得するように設定している時のみ、このイベントの取得を有効にすることができます。

```
onmousemove=関数名、またはスクリプト
```

【利用可能なオブジェクト】

なし

【eventオブジェクトのプロパティ】

| | |
|---|---|
| type | 「mousemove」を値に持つ |
| layerX, layerY, pageX, pageY, screenX, screenY | イベントが発生した時のカーソルのX軸Y軸位置を値に持つ |

## MouseOut

オブジェクトからマウスカーソルが離れた時のイベントを取得します。

```
onmouseout=関数名、またはスクリプト
```

**【利用可能なオブジェクト】**

Area, Layer, Link

**【event オブジェクトのプロパティ】**

| | |
|---|---|
| type | 「mouseout」を値に持つ |
| layerX, layerY, pageX, pageY, screenX, screenY | イベントが発生した時のカーソルのX軸Y軸位置を値に持つ |

## MouseOver

オブジェクトにカーソルが乗った時のイベントを取得します。

```
onmouseover=関数名、またはスクリプト
```

**【利用可能なオブジェクト】**

Area, Layer, Link

**【event オブジェクトのプロパティ】**

| | |
|---|---|
| type | 「mouseover」を値に持つ |
| layerX, layerY, pageX, pageY, screenX, screenY | イベントが発生した時のカーソルのX軸Y軸位置を値に持つ |

## MouseUp

ユーザーがマウスボタンを放した時のイベントを取得します。

もしも「MouseUp」イベントが偽(false)を返した時には、選択やリンクの参照などの通常動作が中止されます。

```
onmouseup=関数名、またはスクリプト
```

**【利用可能なオブジェクト】**

Button, document, Link

**【event オブジェクトのプロパティ】**

| | |
|---|---|
| type | 「mouseup」を値に持つ |
| layerX, layerY, pageX, pageY, screenX, screenY | イベントが発生した時のカーソルのX軸Y軸位置を値に持つ |
| which | 左ボタンの時は「1」を、右ボタンの時は「3」を値に持つ |
| modifiers | イベントが発生した時に押された修飾キーの値を持つ |

## Move

ユーザー、またはスクリプトによって、ウィンドウ、またはフレームが動いた時のイベントを取得します。

```
onmove=関数名、またはスクリプト
```

**【利用可能なオブジェクト】**

window, Frame

**【event オブジェクトのプロパティ】**

| | |
|---|---|
| type | 「move」を値に持つ |
| screenX, screenY | ウィンドウ、またはフレームの左上角の位置の値を持つ |

## Resize

ユーザー、またはスクリプトによって、ウィンドウ、またはフレームのサイズが変更された時のイベントを取得します。

```
onresize=関数名、またはスクリプト
```

**【event オブジェクトのプロパティ】**

| | |
|---|---|
| type | 「resize」を値に持つ |
| width, height | ウィンドウ、またはフレームの幅と高さの値を持つ |

# ナビゲータオブジェクト

## navigatorオブジェクト

ブラウザのユーザーエージェントやアプリケーション名、バージョンなど、ブラウザ固有の情報を提供するオブジェクト。
通常はサーバー側で行われている、ユーザーエージェントなどからブラウザを判断して、そのブラウザに最適化されたWebページを表示させるなどの処理を、ブラウザ側で行うことを可能にします。

```
navigator.property
navigator.method()
```

**【プロパティ】**

(JavaScript1.0)

| | |
|---|---|
| appCodeName | ブラウザのコード名 |
| appName | ブラウザのブラウザ名 |
| appVersion | ブラウザのバージョン |
| userAgent | ブラウザのユーザーエージェント |

(JavaScript1.1)

| | |
|---|---|
| mimeTypes | MIMEタイプ配列(オブジェクト) |
| plugins | プラグイン配列(オブジェクト) |

(JavaScript1.2)

| | |
|---|---|
| language | ユーザーが選択した言語 |
| platform | ユーザーのプラットホームのタイプ |

**【メソッド】**

(JavaScript1.0)

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| javaEnabled() | Javaが使えるか |
| taintEnabled() | 外部からのプロパティ参照を許可しているかどうか |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

**【イベントハンドラ】**

なし

### mimeTypesオブジェクト(配列)(JavaScript1.1 ~)

ブラウザで使用できる使用MIMEタイプの配列を作成する、navigatorオブジェクトのプロパティ。

```
navigator.mimeTypes[インデックス].property
```

**【プロパティ】**

(JavaScript1.0)

未対応

(JavaScript1.1)

| | |
|---|---|
| description | MIMEタイプの詳細 |
| enablePlugin | プラグインを使うための名前 |
| length | MIMEタイプの数 |
| suffixes | MIMEタイプの拡張子 |
| type | MIMEタイプ名 |

**【メソッド】**

(JavaScript1.0)

未対応

(JavaScript1.1)

なし

**【イベントハンドラ】**

なし

### pluginsオブジェクト(配列)(JavaScript1.1 ~)

ブラウザにインストールされている使用可能なプラグインの配列を作成する、navigatorオブジェクトのプロパティ。

```
navigator.plugins[インデックス].property
```

【プロパティ】

(JavaScript1.0)

未対応

(JavaScript1.1)

| | |
|---|---|
| description | プラグインの詳細 |
| filename | プラグインのファイル名 |
| length | プラグインの数 |
| name | プラグインの名前 |

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

| | |
|---|---|
| refresh() | プラグイン配列のリフレッシュ |

【イベントハンドラ】

なし

## screen オブジェクト (JavaScript1.2～)

ディスプレイに関する情報を提供するオブジェクト。

```
screen.property
```

【プロパティ】

(JavaScript1.2)

| | |
|---|---|
| width | ディスプレイの幅(ピクセル) |
| height | ディスプレイの高さ(ピクセル) |
| availWidth | タスクバーなどの部分を除いたディスプレイの幅(ピクセル) |
| availHeight | タスクバーなどの部分を除いたディスプレイの高さ(ピクセル) |
| pixelDepth | ディスプレイ深度(bits/pixel) |
| colorDepth | ディスプレイ色深度(nビットカラー) |

【メソッド】

なし

【イベントハンドラ】

なし

## event オブジェクト (JavaScript1.2～)

イベントの情報を取り扱うオブジェクト。

タグ内に設定されたイベントハンドラからだけでなく、ウィンドウ上のどこからでもイベントを取得することができます。

```
event.property
```

【プロパティ】

(JavaScript1.2)

| | |
|---|---|
| type | イベントのタイプ |
| layerX | イベントが発生したレイヤー上のX座標(ピクセル) |
| layerY | イベントが発生したレイヤー上のY座標(ピクセル) |
| pageX | イベントが発生したページ上のX座標(ピクセル) |
| pageY | イベントが発生したページ上のY座標(ピクセル) |
| screenX | イベントが発生したディスプレイ上のX座標(ピクセル) |
| screenY | イベントが発生したディスプレイ上のY座標(ピクセル) |
| which | マウスボタンが押されたり、押されたキーのASCIIの値の数字を表す |
| modifiers | マウスまたはキーイベント発生時のモディファイキー(修飾キー)の種類(ALT_MASK, CONTROL_MASK, SHIFT_MASK, META_MASK) |
| data | DragDropイベント発生時の、ドロップされたオブジェクトのURL |

【メソッド】

なし

【イベントタイプ】

すべてのイベントハンドラおよびイベントタイプ

## windowオブジェクト

ブラウザ自身や、アラートウィンドウなどの各種ダイアログボックスの情報を提供したり、操作を行うオブジェクト。

documentオブジェクト、Frameオブジェクト、historyオブジェクト、locationオブジェクトなど、多くのオブジェクトを含んでいます。

最上層のオブジェクトなので「window」は省略可能ですが、「open()」メソッドや「close()」メソッドなど、他のオブジェクトと同じ名称のコマンドもあるので、混乱を避けるためには「window」を記述したほうがよいでしょう。

JavaScript1.2から、ウィンドウ自身のサイズや表示位置を、より細かく設定できるようになりました。

```
window.property
window.method()
self.property
self.method()
top.property
top.methodN()
parent.property
parent.method()
windowVar.property
windowVar.methodName()
property
method()
```

**【プロパティ】**

(JavaScript1.0)

| | |
|---|---|
| defaultStatus | デフォルトのステータス行のメッセージ |
| length | ウィンドウ内のフレーム数 |
| name | ウィンドウ名 |
| parent | ウィンドウ内のフレームの親フレーム |
| self | 現在のウィンドウ自身。「window」と同じ |
| status | ステータス行のメッセージ |
| top | 1番手前にあるウィンドウ |
| window | 現在のウィンドウ自身。「self」と同じ |
| document | documentオブジェクト |
| Frame(s) | Frameオブジェクトおよび配列 |
| history | historyオブジェクトおよび配列 |
| location | locationオブジェクト |

(JavaScript1.1)

| | |
|---|---|
| closed | ウィンドウが閉じられている状態 |
| opener | 「open()」メソッドで開かれたウィンドウから元のウィンドウを参照する |

(JavaScript1.2)

| | |
|---|---|
| innerHeight | ウィンドウの表示領域の高さ(ピクセル) |
| innerWidth | ウィンドウの表示領域の幅(ピクセル) |
| locationbar | ロケーションバー |
| menubar | ウィンドウのメニューバー |
| outerHeight | ウィンドウの外周の高さ(ピクセル) |
| outerWidth | ウィンドウの外周の幅(ピクセル) |
| pageXOffset | ウィンドウのX座標の位置 |
| pageYOffset | ウィンドウのY座標の位置 |
| personalbar | ウィンドウのパーソナルバー |
| scrollbars | ウィンドウのスクロールバー |
| statusbar | ウィンドウのステータスバー |
| toolbar | ウィンドウのツールバー |

**【メソッド】**

(JavaScript1.0)

| | |
|---|---|
| alert() | 警告用ダイアログボックスを開く |
| clearTimeout() | 「setTimeout()」メソッドの停止 |
| close() | ウィンドウを閉じる |
| confirm() | 確認ボタン付きダイアログボックスを開く |
| open() | 新しいウィンドウを開く |
| prompt() | 入力欄付きのダイアログボックスを開く |
| setTimeout() | ミリ秒単位で指定した時間後に実行する |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| scroll() | ウィンドウをスクロールする |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| back() | ひとつ前のURLに進む |
| disableExternal Capture | 外部からのイベントキャプチャを無効にする |
| enableExternal Capture | 異なったロケーション(サーバー)からのイベントキャプチャを許す(Signed Scriptなどと合わせて使用する) |
| find() | ウィンドウの中に指定した文字列があるか(あればtrue、なければfalse) |
| forward() | ひとつ先のURLに進む |
| home() | homeに指定したURLに進む |
| moveBy() | ウィンドウを移動する(水平方向と垂直方向の移動量をピクセルで指定) |
| moveTo() | ウィンドウを移動する(ウィンドウの左上の角を基準にピクセルで指定) |
| resizeBy() | ウィンドウをリサイズする(底辺の角を基準に水平方向と垂直方向をピクセルで指定) |
| resizeTo() | ウィンドウをリサイズする(ウィンドウのサイズをピクセルで指定) |
| scrollBy() | ウィンドウをスクロールする(ウィンドウの表示領域の水平方向と垂直方向に対してピクセルで指定) |
| scrollTo() | ウィンドウをスクロールする(ウィンドウの左上の角を基準にピクセルで指定) |
| stop() | 読み込みを中止する |
| captureEvents() | すべてのタイプのイベントを取得する |
| setInterval() | 一定時間(1000分の1秒)ごとに指定された処理を繰り返す |
| clearInterval() | 「setInterval()」メソッドの停止 |
| handleEvent() | イベント取り扱い者を特定する |
| print() | プリントする |
| releaseEvents() | 別階層のイベントにイベントを渡す |
| routeEvent() | 取得したイベントと同じ階層のイベント |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

**【open()の属性オプション】**

(JavaScript1.0)

＊：[=yes¦no]¦[=1¦0]

| | |
|---|---|
| toolbar * | ツールバー |
| location * | ロケーションボックス |
| directories * | ディレクトリボタン |
| status * | ステータスバー |
| menubar * | メニューバー |
| scrollbars * | スクロールバー |
| resizable * | リサイズボックス |
| width=pixels | ウィンドウの横幅 |
| height=pixels | ウィンドウの縦幅 |

(JavaScript1.2)

＊：[=yes¦no]¦[=1¦0]

| | |
|---|---|
| always Lowered * | 他のウィンドウより前に来るウィンドウ |
| always Raised * | 他のウィンドウより後ろに来るウィンドウ |
| dependent * | 現在のウィンドウの子ウィンドウ |
| hotkeys * | ホットキーを無力にする |
| innerWidth =pixels | ウィンドウの表示領域の横幅(widthから変更) |
| innerHeight =pixels | ウィンドウの表示領域の高さ(heightから変更) |
| outerWidth =pixels | ウィンドウの外周の幅 |
| outerHeight =pixels | ウィンドウの外周の高さ |
| screenX =pixels | ディスプレイ左上からのX軸の位置 |
| screenY =pixels | ディスプレイ左上からのY軸の位置 |
| titlebar * | タイトルバーの表示 |
| z-lock * | フォーカスが移っても他のウィンドウの前に来ないウィンドウ |

**【イベントハンドラ】**

(JavaScript1.0)

onLoad

onUnload

(JavaScript1.1)

onBlur

onError

onFocus

## frame オブジェクト

frame オブジェクトの情報を提供したり、操作を行うオブジェクト。

frame オブジェクトは、それ自体が window オブジェクトのプロパティですが、最上層のオブジェクトである「window」は省略可能です。

ウィンドウ内のフレームの配列を作成します。フレームの情報を取得して利用することによって、フレーム操作の幅を広げることができます。

```
frame名.property
frames[インデックス].property
window.property
self.property
parent.property
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| frames | フレーム配列 |
| name | フレーム名 |
| length | フレームの数 |
| parent | 現在のフレームの親フレーム |
| self | 現在のフレーム自身 |
| window | 現在のフレームのウィンドウ自身 |

(JavaScript1.1)

なし

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| clearTimeout() | 「setTimeout()」メソッドの停止 |
| setTimeout() | ミリ秒(1000分の1秒)単位で指定された時間後に命令を実行する |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| setInterval() | 一定時間(1000分の1秒)ごとに指定された処理を繰り返す |
| clearInterval() | 「setInterval()」メソッドの停止 |
| handleEvent() | イベント取り扱い者を特定する |
| print() | プリントする |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

(JavaScript1.0)

なし

(JavaScript1.1)

onblur

onFocus

## document オブジェクト

HTML ファイル、およびそこに記述されるドキュメントの情報を提供したり、操作を行うオブジェクト。

Layer オブジェクト、Link オブジェクト、Image オブジェクト、Area オブジェクト、Anchor オブジェクト、Applet オブジェクト、Form オブジェクトなど、多くのオブジェクトを含んでいます。

document オブジェクトは、それ自体が window オブジェクトのプロパティですが、最上層のオブジェクトである「window」は省略可能です。

```
(window.)document.property
(window.)document.method()
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| alinkColor | アクティブリンクの色 |
| bgColor | 背景の色 |
| cookie | クッキーファイルの情報 |
| fgColor | テキストの色 |
| lastModified | ファイルの最終更新日時 |
| linkColor | リンクの色 |
| referrer | リンク元のURL |
| title | ドキュメントのタイトル |
| URL | カレントのURL |
| vlinkColor | すでに行ったことのあるリンクの色 |

| | |
|---|---|
| anchor(s) | アンカーオブジェクトおよび配列 |
| form(s) | フォームオブジェクトおよび配列 |
| link(s) | リンクオブジェクトおよび配列 |

(JavaScript1.1)

| | |
|---|---|
| domain | カレントのドメイン名(tainting状態時のみ使用可能) |
| applet(s) | アプレット配列 |
| Area | エリアオブジェクト |
| embeds | プラグイン配列 |
| image(s) | イメージオブジェクトおよび配列 |

(JavaScript1.2)

| | |
|---|---|
| layer(s) | レイヤーオブジェクトおよび配列 |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| close() | ドキュメントストリームを閉じる |
| open() | ドキュメントストリームを開く |
| write() | テキストを書き出す |
| writeln() | テキストを改行付きで書き出す |
| toString() | オブジェクトを文字列にする |

(JavaScript1.1)

| | |
|---|---|
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| getSelection() | 選択範囲内に含まれている文字列を返す |
| capture Events() | すべてのタイプのイベントを取得する |
| release Events() | 別階層のイベントにイベントを渡す |
| routeEvent() | 取得したイベントと同じ階層のイベント |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

なし

【JavaScript1.1で削除されたプロパティ・メソッド】

location

clear()

## historyオブジェクト

ブラウザがロードしたページの来歴を提供するオブジェクト。

historyオブジェクトは、それ自体がwindowオブジェクトのプロパティですが、最上層のオブジェクトである「window」は省略可能です。

```
(window.)history.property
(window.)history.method()
(window.)history[インデックス]
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| length | 来歴の数 |

### 注意

### 削除されたプロパティ・メソッドについて

現在、Netscape社の「JavaScript Guide」では、JavaScript1.0にはあった「location」プロパティと「clear()」メソッドが姿を消しています。

「location」プロパティは、以前より「locationプロパティは将来的には使えなくなるので、URLプロパティを使うように」とのアナウンスがされており、これはおそらくlocationオブジェクトとの混乱を避けるための処置だと思われます。

また、「clear()」メソッドに関しては、「open()」メソッドなどを利用して似たような効果を出すことが可能なため、削除されたものと思われます。

なお、このふたつのコマンドは、今のところNetscape Navigator 3.0で使用可能なようですが(Windows版では「clear()」メソッドは正常に動かない可能性があります)、今後のことを考えるとなるべく使用しないことをお勧めします。

(JavaScript1.1)

| | |
|---|---|
| current | 現在の来歴（tainting状態時のみ使用可能） |
| next | 次の来歴（tainting状態時のみ使用可能） |
| previous | ひとつ前の来歴（tainting状態時のみ使用可能） |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| back() | ひとつ前のページに進む |
| forward() | ひとつ先のページへ進む |
| go() | 指定された分だけページを移動する |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| valueOf() | オブジェクトの値を返す |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

なし

## locationオブジェクト

表示されているページのURLに関する情報を提供するオブジェクト。

locationオブジェクトは、それ自体がwindowオブジェクトのプロパティですが、最上層のオブジェクトである「window」は省略可能です。

また、locationオブジェクトは読み出しだけでなく、動的にURLを変更することが可能です。

```
(window.)location.property
(window.)location.method()
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| hash | アンカー名 |
| host | URLのホスト名とポート番号部分 |
| hostname | URLのホストコンピュータ名部分 |
| href | URL |
| pathname | URLのパス名部分 |
| port | URLのポート番号部分 |
| protocol | URLのプロトコル部分 |
| search | URLの問い合わせ部分 |

(JavaScript1.1)

なし

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| reload() | リロードする |
| replace() | 現在のURLを置き換える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

なし

## Linkオブジェクト

HTMLファイル内のリンクの配列を作成し、リンクひとつひとつの情報を提供するオブジェクト。

Linkオブジェクトは、それ自体がdocumentオブジェクトのプロパティです。

```
document.links[インデックス].property
document.links.length
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| hash | アンカー名 |
| host | リンク先のURLのホスト名とポート番号部分 |
| length | リンク数 |
| hostname | リンク先のURLのホストコンピュータ名部分 |
| href | リンク先のURL |
| pathname | リンク先のURLのパス名部分 |
| port | リンク先のURLのポート番号部分 |
| protocol | リンク先のURLのプロトコル部分 |
| search | リンク先のURLの問い合わせ部分 |
| target | リンク先のターゲット属性 |

(JavaScript1.1)
なし

【メソッド】
(JavaScript1.0)

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】
(JavaScript1.0)
onClick
onMouseOver

(JavaScript1.1)
onMouseOut

## Anchorオブジェクト(配列)

HTMLファイル内のAnchorの配列を作成するオブジェクト。
Anchorオブジェクトは、それ自体がdocumentオブジェクトのプロパティです。

```
document.anchors[インデックス]
document.anchors.length
```

【プロパティ】
(JavaScript1.0)

| | |
|---|---|
| length | アンカーの数 |

(JavaScript1.1)
なし

【メソッド】
(JavaScript1.0)

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| valueOf() | オブジェクトの値を返す |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】
なし

## Formオブジェクト

フォームに関する情報を提供したり、操作を行うオブジェクト。
Formオブジェクトは、それ自体がdocumentオブジェクトのプロパティです。
HTMLの<form>を、JavaScriptのオブジェクトとして取り扱います。データ入力を受け付けて送信するだけでなく、入力内容のチェックや入力されたデータの計算をブラウザ側で行ったり、リアルタイムでフォームの内容を変化させるなど、フォームの利用方法を広げることが可能です。
Textareaオブジェクト、Textオブジェクト、FileUploadオブジェクト、Passwordオブジェクト、Hiddenオブジェクト、Submitオブジェクト、Resetオブジェクト、Radioオブジェクト、Checkboxオブジェクト、Buttonオブジェクト、Selectオブジェクトなど、多くのオブジェクトを含んでいます。

```
form名.property
form名.method(数値)
forms[インデックス].property
forms[インデックス].method()
```

【プロパティ】
(JavaScript1.0)

| | |
|---|---|
| action | フォーム内のデータの送り先 |
| elements | フォーム内の内容(配列) |
| encoding | フォームのMINEエンコード |
| length | フォームエレメントの数 |
| name | オブジェクト名 |
| method | フォームの送信方法 |
| target | ターゲットウィンドウ |
| Button | Buttonオブジェクト |
| Checkbox | Checkboxオブジェクト |
| Hidden | Hiddenオブジェクト |

| | |
|---|---|
| Password | Passwordオブジェクト |
| Radio | Radioオブジェクト |
| Reset | Resetオブジェクト |
| Select | Selectオブジェクト |
| Submit | Submitオブジェクト |
| Text | Textオブジェクト |
| Textarea | Textareaオブジェクト |

(JavaScript1.1)

| | |
|---|---|
| FileUpload | FileUploadオブジェクト |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| submit() | 送信ボタンが押されたのと同じ |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| reset() | リセットボタンが押された状態と同じ |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

(JavaScript1.0)

onSubmit

(JavaScript1.1)

onReset

## Buttonオブジェクト

フォームのButtonに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<input type="button">を使って作成されたボタン型のフォームを、JavaScriptのオブジェクトとして取り扱います。

```
button名.property
button名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| name | オブジェクト名 |
| value | オブジェクト内の値 |

(JavaScript1.1)

| | |
|---|---|
| type | オブジェクトのタイプ |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| click() | クリックと同じ |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

(JavaScript1.0)

onClick

(JavaScript1.1)

onBlur

onFocus

## Checkboxオブジェクト

フォームのCheckboxに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<input type="Checkbox">を使って作成されたチェックボックス型のフォームを、JavaScriptのオブジェクトとして取り扱います。

```
checkbox名.property
checkbox名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

（JavaScript1.0）

| | |
|---|---|
| checked | チェックされている状態 |
| defaultChecked | デフォルトでチェックされている状態 |
| name | オブジェクト名 |
| value | オブジェクト内の値 |

（JavaScript1.1）

| | |
|---|---|
| type | オブジェクトのタイプ |

【メソッド】

（JavaScript1.0）

| | |
|---|---|
| click() | クリックと同じ |
| toString() | オブジェクトを文字列に変える |

（JavaScript1.1）

| | |
|---|---|
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| valueOf() | オブジェクトの値を返す |

（JavaScript1.2）

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

（JavaScript1.3）

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

（JavaScript1.0）

onClick

（JavaScript1.1）

onBlur

onFocus

## FileUploadオブジェクト(JavaScript1.1~)

フォームのFileUploadに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<input type="FileUpload">を使って作成されたファイル選択欄型のフォームを、JavaScriptのオブジェクトとして取り扱います。

```
FileUpload名.property
FileUpload名.method()
```

【プロパティ】

（JavaScript1.0）

未対応

（JavaScript1.1）

| | |
|---|---|
| name | オブジェクト名 |
| value | オブジェクト内の値 |
| type | オブジェクトのタイプ |

【メソッド】

（JavaScript1.0）

未対応

（JavaScript1.1）

| | |
|---|---|
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| toString() | オブジェクトを文字列に変える |
| valueOf() | オブジェクトの値を返す |

（JavaScript1.2）

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

（JavaScript1.3）

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

（JavaScript1.0）

未対応

（JavaScript1.1）

onBlur

onChange

onFocus

## Hiddenオブジェクト

フォームのHiddenに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<input type="Hidden">を使って作成された隠しテキストボックス型のフォームを、JavaScriptのオブジェクトとして取り扱います。

```
Hidden名.property
form名.elements[インデックス].property
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| name | オブジェクト名 |
| value | オブジェクト内の値 |

(JavaScript1.1)

| | |
|---|---|
| type | オブジェクトのタイプ |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

なし

## Passwordオブジェクト

フォームのPasswordに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<input type="Password">を使って作成されたパスワード入力用テキストボックス型のフォームを、JavaScriptのオブジェクトとして取り扱います。

```
Password名.property
Password名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| defaultValue | デフォルトのオブジェクト内の値 |
| name | オブジェクト名 |
| value | オブジェクト内の値 |

(JavaScript1.1)

| | |
|---|---|
| type | オブジェクトのタイプ |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| select() | 選択された状態 |
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

(JavaScript1.0)

なし

(JavaScript1.1)

onBlur

onFocus

## Radioオブジェクト

フォームのRadioに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<input type="Radio">を使って作成されたラジオボタン型のフォームを、JavaScriptのオブジェクトとして取り扱います。

```
Radio名[インデックス].property
Radio名[インデックス].method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| checked | 選択されている状態 |
| defaultChecked | デフォルトで選択されている状態 |
| length | オブジェクトの数 |
| name | オブジェクト名 |
| value | オブジェクト内の値 |

(JavaScript1.1)

| | |
|---|---|
| type | オブジェクトのタイプ |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| click() | クリックと同じ |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

(JavaScript1.0)

onClick

(JavaScript1.1)

onBlur

onFocus

## Resetオブジェクト

フォームのResetに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<input type="Reset">を使って作成されたリセットボタン型のフォームを、JavaScriptのオブジェクトとして取り扱います。

```
Reset名[インデックス].property
Reset名[インデックス].method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| name | オブジェクト名 |
| value | オブジェクト内の値 |

(JavaScript1.1)

| | |
|---|---|
| type | オブジェクトのタイプ |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| click() | クリックと同じ |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

(JavaScript1.0)

onClick

(JavaScript1.1)

onBlur

onFocus

## Selectオブジェクト

フォームのSelectに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<select>と<option>を使って作成された選択欄型のフォームと、その中の選択項目を、JavaScriptのオブジェクトとして取り扱います。

【用法】

```
select名[インデックス].property
select名[インデックス].method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【オプション】

```
select名.options[インデックス].property
form名.elements[インデックス].options
[インデックス].property
option名.property
```

リファレンス

【オプション配列】

```
select名.options
select名.options[インデックス]
select名.options.length
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| length | オプションの数 |
| name | オブジェクト名 |
| options | オプション項目 |
| selectedIndex | 選択されている項目 |

(JavaScript1.1)

| | |
|---|---|
| type | オブジェクトのタイプ |

【オプション(配列)】

(JavaScript1.0)

| | |
|---|---|
| defaultSelected | デフォルトで選択されている項目 |
| index | 項目の位置 |
| selected | 選択されている状態 |
| value | オブジェクト内の値 |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| click() | クリックと同じ |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

(JavaScript1.0)

onChange

(JavaScript1.1)

onBlur

onFocus

## Submitオブジェクト

フォームのSubmitに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<input type="Submit">を使って作成されたフォーム内容送信用ボタン型のフォームを、JavaScriptのオブジェクトとして取り扱います。

```
Submit名.property
Submit名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| name | オブジェクト名 |
| value | オブジェクト内の値 |

(JavaScript1.1)

| | |
|---|---|
| type | オブジェクトのタイプ |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| click() | クリックと同じ |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

(JavaScript1.0)

onClick

(JavaScript1.1)

onBlur

onFocus

## Textareaオブジェクト

フォームのTextareaに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<textarea>使って作成されたテキストエリア型のフォームを、JavaScriptのオブジェクトとして取り扱います。

```
textarea名.property
textarea名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| defaultValue | デフォルトのオブジェクトの値 |
| name | オブジェクト名 |
| value | オブジェクト内の値 |

(JavaScript1.1)

| | |
|---|---|
| type | オブジェクトのタイプ |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

(JavaScript1.0)

onBlur
onChange
onFocus
onSelect

(JavaScript1.1)

なし

## Textオブジェクト

フォームのTextに関する情報を提供したり、操作を行うオブジェクト。

HTMLの<input type="Text">を使って作成されたテキスト入力欄型のフォームを、JavaScriptのオブジェクトとして取り扱います。

```
Text名.property
Text名.method()
form名.elements[インデックス].property
form名.elements[インデックス].method()
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| defaultValue | デフォルトのオブジェクトの値 |
| name | オブジェクト名 |
| value | オブジェクト内の値 |

(JavaScript1.1)

| | |
|---|---|
| type | オブジェクトのタイプ |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| blur() | フォーカスを移動する |
| focus() | フォーカスを与える |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

(JavaScript1.0)

onBlur
onChange
onFocus
onSelect

(JavaScript1.1)

なし

## Areaオブジェクト (JavaScript1.1)

イメージマップの定義や、その情報を提供するオブジェクト。
Areaオブジェクトは、Linkの配列内にあります。したがって、リンクを参照する場合は、「document.links[インデックス]」を使用します。

```
area名.property
document.links[インデックス].property
```

【プロパティ】
(JavaScript1.0)
未対応

(JavaScript1.1)

| | |
|---|---|
| hash | アンカー名 |
| host | リンク先のURLのホスト名とポート番号部分 |
| hostname | リンク先のURLのホストコンピュータ名部分 |
| href | リンク先のURL |
| pathname | リンク先のURLのパス名部分 |
| port | リンク先のURLのポート番号部分 |
| protocol | リンク先のURLのプロトコル部分 |
| search | リンク先のURLの問い合わせ部分 |
| target | リンク先のターゲット属性 |

【メソッド】
(JavaScript1.0)
未対応

(JavaScript1.1)

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】
(JavaScript1.0)
未対応

(JavaScript1.1)
onMouseOver
onMouseOut

## Imageオブジェクト (JavaScript1.1～)

画像に関する情報を提供したり、操作を行うオブジェクト。
画像ファイルの情報を提供する以外に、画像を後から置き換えることも可能です。これにより、画像をアニメーションさせたり、イベントハンドラと組み合わせることによって、インタラクティブに画像を取り扱うことができます。
ビルトインオブジェクトのように、new演算子を使って、オブジェクトを新たに作成することもできます。

【オブジェクトの作成】

```
image名 = new Image(画像の幅, 画像の高さ)
```

【用法】

```
オブジェクト名.property
document.images[インデックス].property
```

【プロパティ】
(JavaScript1.0)
未対応

(JavaScript1.1)

| | |
|---|---|
| border | borderで設定された値 |
| complete | 画像の読み込みが終了しているかどうか |
| height | イメージの高さ |
| hspace | heightで設定された値 |
| length | イメージの数 |
| lowsrc | lowsrcで設定している画像ファイルのURL |
| name | イメージの名前 |
| prototype | 新しいプロパティの作成 |
| src | 画像ファイルのURL |
| vspace | vspaceで設定された値 |
| width | 画像の幅 |

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| handleEvent() | イベント取り扱い者を特定する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

(JavaScript1.1)

onAbort

onError

onLoad

## Layer オブジェクト (JavaScript1.2～)

レイヤーに関する情報を提供したり、操作を行うオブジェクト。

Netscape Navigator 4.0 で追加された <layer> に関連した JavaScript です。

レイヤーの show(見える状態)と hide(隠された状態)の動的な切り替えや、指定位置への移動など、インタラクティブな要素を付け加えることができます。

Layer オブジェクトは、document オブジェクトのプロパティです。

【用法】

```
layer名.property
layer名.method(数値)
document.layer名
document.layers[インデックス]
document.layers["layer名"]
```

【ネストされているレイヤーのプロパティを使う時】

```
document.layers["親layer"].layers[
"子layer"]
```

【プロパティ】

(JavaScript1.2)

| | |
|---|---|
| name | オブジェクト名の設定 |
| width | レイヤーのコンテンツの横幅(ピクセル) |
| height | レイヤーのコンテンツの高さ(ピクセル) |
| left | 画面左からのレイヤーの位置(ピクセル) |
| top | 画面上からのレイヤーの位置(ピクセル) |
| visibility | レイヤーがshow(見える状態)かhide(隠された状態)かを返す |
| zIndex | レイヤーのZ-Indexの値を返す |
| pageX | ページ上のX座標(ピクセル) |
| pageY | ページ上のY座標(ピクセル) |
| x | 「Layer.left.」と同義 |
| y | 「Layer.top.」と同義 |
| clip.top | レイヤー内のクリップの上からの位置 |
| clip.left | レイヤー内のクリップの左からの位置 |
| clip.right | レイヤー内のクリップの右からの位置 |
| clip.bottom | レイヤー内のクリップの下からの位置 |
| clip.width | レイヤー内のクリップの横幅 |
| clip.height | レイヤー内のクリップの高さ |
| background | レイヤーの背景画像の指定 |
| bgColor | レイヤーの背景色の指定 |
| siblingAbove | 兄弟レイヤー内でひとつ上のレイヤーを参照する。もしも1番先頭だったら「null」を返す |
| siblingBelow | 兄弟レイヤー内でひとつ下のレイヤーを参照する。もしも1番後ろだったら「null」を返す |
| above | ひとつ上のレイヤーを参照する。もしも1番先頭だったら「null」を返す |
| below | ひとつ下のレイヤーを参照する。もしも1番後ろだったら「null」を返す |
| parentLayer | 符合するレイヤーの状況。親のレイヤーだったら「null」を返す |

| | |
|---|---|
| layers | LAYER配列 |
| src | レイヤーのソースを外部から呼び出す時のURL |

【メソッド】

(JavaScript1.2)

| | |
|---|---|
| moveBy() | 現在の表示位置からのレイヤーの表示位置移動(ピクセル) |
| moveTo() | 画面の左上からのレイヤーの表示位置移動(ピクセル) |
| moveTo Absolute() | レイヤーコンテンツ内でのレイヤー位置移動(ピクセル) |
| resizeBy() | レイヤーのサイズを指定された分だけ変更する(ピクセル) |
| resizeTo() | レイヤーのサイズを指定されたサイズへ変更する(ピクセル) |
| moveAbove() | レイヤーを前に出す |
| moveBelow() | レイヤーを後ろに入れる |
| load() | 外部からレイヤーのHTMLファイルを読み込む |
| captureEvents() | すべてのタイプのイベントを取得する |
| handleEvent() | イベント取り扱い者を特定する |
| releaseEvents() | 別階層のイベントにイベントを渡す |
| routeEvent() | 取得したイベントと同じ階層のイベント |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

onMouseOver

onMouseOut

onLoad

onFocus

onBlur

## Appletオブジェクト(配列)(JavaScript1.1 ~)

JAVAアプレット関連のオブジェクトおよび配列。

```
document.applets[インデックス]
document.applets.length
```

【プロパティ】

(JavaScript1.1)

| | |
|---|---|
| length | アプレットの数(配列数) |

【メソッド】

(JavaScript1.1)

JAVAアプレットのすべての「public method」

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |
| valueOf() | オブジェクトの値を返す |

## embeds配列(JavaScript1.1 ~)

プラグイン関連のオブジェクトおよび配列。

```
document.embeds[インデックス]
document.embeds.length
```

【プロパティ】

(JavaScript1.1)

| | |
|---|---|
| length | プラグインの数(配列数) |

【メソッド】

(JavaScript1.1)

各プラグイン独自のメソッド

# ビルトインオブジェクト

## Date オブジェクト

日付や時間の取得や設定を行うオブジェクト。ブラウザの起動時に、そのマシンから時間の情報を取得して加工します。各マシンが置かれている場所のローカルタイムを扱うので、CGIでは難しい世界各地のローカルタイムに合わせた処理を行うことができます。

**【オブジェクトの作成】**

```
オブジェクト名 = new Date()
オブジェクト名 = new Date("月 日, 年 時:分:秒")
オブジェクト名 = new Date(年, 月, 日)
オブジェクト名 = new Date(年, 月, 日, 時, 分, 秒)
```

(JavaScript1.3)

```
オブジェクト名 = new Date (年,月,日, [,時[,分[,秒[,ミリ秒]]]])
オブジェクト名 = new Date.UTC(年, 月, 日, [,時[,分[,秒[, ミリ秒]]]])
```

**【用法】**

```
オブジェクト名.property
オブジェクト名.method()
```

**【プロパティ】**

(JavaScript1.0)

なし

(JavaScript1.1)

| | |
|---|---|
| constructor | オブジェクトのプロトタイプの作成元 |
| prototype | プロパティの追加 |

**【メソッド】**

(JavaScript1.0)

| | |
|---|---|
| getDate() | 日を返す(1～31) |
| getDay() | 曜日を返す(0～6) |
| getHours() | 時間を返す(0～23) |
| getMinutes() | 分を返す(0～59) |
| getMonth() | 月を返す(0～11) |
| getSeconds() | 秒を返す(0～59) |
| getTime() | 1970年1月1日0時0分0秒からの経過時間をミリ秒単位で返す |
| getTimezoneOffset() | グリニッジ標準時(GTM)とローカル時間の差を分単位で返す |
| getYear() | 年を返す |
| parse() | 指定された日付と時間と1970年1月1日0時0分0秒との間の時間をミリ単位で返す |
| setDate() | 日付の設定をする |
| setHours() | 時間の設定をする |
| setMinutes() | 分の設定をする |
| setMonth() | 月の設定をする |
| setSeconds() | 秒の設定をする |
| setTime() | ミリ秒単位で日付と時間の設定をする |
| setYear() | 年の設定をする |
| toGMTString() | 日付と時間をグリニッジ標準時(GTM)の文字列に変換する |
| toLocaleString() | 日付けと時間をローカルタイムの文字列で返す |
| UTC() | 1970年1月1日0時0分0秒からの経過時間をグリニッジ標準時(GTM)を元にミリ秒単位で返す |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| valueOf() | オブジェクトの値を返す |

(JavaScript1.3)

| | |
|---|---|
| getFullYear() | 4桁の年を返す |
| setFullYear() | 4桁の年を設定する |
| getMilliseconds() | ミリセコンドを返す(0～999) |
| setMilliseconds() | ミリセコンドを設定する |
| getUTCFullYear() | 4桁のUTC時間の年を返す |

| | |
|---|---|
| getUTCMonth() | UTC時間の月を返す(0～11) |
| getUTCDate() | UTC時間の日を返す(1～31) |
| getUTCDay() | UTC時間の曜日を返す(0～6) |
| getUTCHours() | UTC時間の時間を返す(0～23) |
| getUTCMinutes() | UTC時間の分を返す(0～59) |
| getUTCSeconds() | UTC時間の秒を返す(0～59) |
| getUTCMilliseconds() | UTC時間のミリセコンドを返す(0～999) |
| toUTCString() | UTC時間を文字列で返す |
| setUTCFullYear() | 4桁のUTC時間の年を設定する |
| setUTCMonth() | 4桁のUTC時間の月を設定する |
| setUTCDate() | 4桁のUTC時間の日を設定する |
| setUTCHours() | UTC時間の時間を設定する |
| setUTCMinutes() | UTC時間の分を設定する |
| setUTCSeconds() | UTC時間の秒を設定する |
| setUTCMilliseconds() | UTC時間のミリセコンドを設定する |
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

なし

## Mathオブジェクト

数値計算関連のオブジェクト。「Math」の後に利用するプロパティやメソッドを記述して使用します。それらを組み合わせて使うことによって、サーバーにデータを渡すことなく、いろいろな計算をブラウザで行うことができます。

```
Math.property
Math.method()
```

【プロパティ】

(JavaScript1.0)

| | |
|---|---|
| E | 自然対数の底 |
| LN2 | eを底とする2の自然対数 |
| LN10 | eを底とする10の自然対数 |
| LOG2E | 2を底とする自然対数 |
| LOG10E | 10を底とする自然対数 |
| PI | 円周率 |
| SQRT1_2 | 1/2の平方根 |
| SQRT2 | 2の平方根 |

(JavaScript1.1)

なし

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| abs() | 0からの絶対値を返す |
| acos() | アークコサインを返す |
| asin() | アークサインを返す |
| atan() | アークタンジェントを返す |
| atan2() | 座標から角度を返す |
| ceil() | もっとも近くて大きい整数を返す |
| cos() | コサインを返す |
| exp() | 対数を返す |
| floor() | もっとも近くて小さい整数を返す |
| log() | 自然対数を返す |
| max() | 比較して大き[illegible]の数値を返す |
| min() | 比較して小さい方の数値を返す |
| pow() | 乗算をする |
| random() | 乱数を発生する(Netscape Navigator 2.XはUNIX版のみ) |
| round() | 四捨五入する |
| sin() | サインを返す |
| sqrt() | 平方根を返す |
| tan() | タンジェントを返す |
| toString() | オブジェクトを文字列にする |

(JavaScript1.1)

| | |
|---|---|
| valueOf() | オブジェクトの値を返す |

【イベントハンドラ】

なし

## stringオブジェクト

文字列に対して修飾や検索などの操作を行うオブジェクト。文字列の後にプロパティやメソッドを付けて使用します。

JavaScript1.1から、new演算子を使って新しくオブジェクトを作成することができるようになりました。

**【オブジェクトの作成】**

(JavaScript1.1)

```
オブジェクト名 = new String(文字列)
```

**【用法】**

```
文字列.property
文字列.method()
```

**【プロパティ】**

(JavaScript1.0)

| | |
|---|---|
| length | 文字の数 |

(JavaScript1.1)

| | |
|---|---|
| constructor | オブジェクトの作成元 |
| prototype | オブジェクトのプロトタイプを作成 |

**【メソッド】**

(JavaScript1.0)

| | |
|---|---|
| anchor() | アンカーを設定する |
| big() | 文字を大きくする |
| blink() | 文字を点滅させる |
| bold() | 太文字にする |
| charAt() | 文字を抜き出す |
| fixed() | 等幅文字にする |
| fontcolor() | 文字色を指定する |
| fontsize() | フォントサイズを指定する |
| indexOf() | 文字を検索する |
| italics() | 斜体文字にする |
| lastIndexOf() | 文字列の後ろから検索する |
| link() | リンクを作成する |
| small() | 文字を小さくする |
| strike() | 削除文字にする |
| sub() | 下付文字にする |
| substring() | 文字列の途中を抜き出す |
| sup() | 上付文字にする |
| toLowerCase() | 大文字を小文字にする |
| toUpperCase() | 小文字を大文字にする |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| split() | 文字列を分割する |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| charCodeAt() | ISO-Latin-1のコード番号を返す |
| fromCharCode() | ISO-Latin-1のコード番号を文字に直す |
| substr() | 文字を抜き出す(n番からm個の文字) |
| match() | 指定したパターンと同じパターンを見つける |
| replace() | 指定したパターンを置き換える |
| split() | 指定したパターンの部分で文字列を分ける |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

**【イベントハンドラ】**

なし

## Arrayオブジェクト (JavaScript1.1)

配列を作成し、操作するオブジェクト。複雑な手順なしに配列が作成できます。

JavaScript1.1から追加されたオブジェクトです。一部のNetscape Navigator 2.0から利用可能ですが、正式なサポートではありません。

**【オブジェクトの作成】**

```
arrayオブジェクト名 = new Array(配列の数)
arrayオブジェクト名 = new Array(0番目の要素,1番目の要素, ...,n番目の要素)
```

**【用法】**

```
オブジェクト名.property
オブジェクト名.method()
```

**【プロパティ】**

(JavaScript1.0)

未対応

リファレンス

(JavaScript1.1)

| | |
|---|---|
| length | 配列の数 |
| constructor | オブジェクトの作成元 |
| prototype | 配列のプロトタイプ |

(JavaScript1.2)

| | |
|---|---|
| index | 正規表現の一致によって作成された配列のための、一致した文字列の0を基準としたインデックス |
| input | 正規表現の一致によって作成された配列のための、正規表現が一致したオリジナルの文字列を反映 |

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

| | |
|---|---|
| join() | 要素を文字列にする |
| reverse() | 要素の順番を逆にする |
| sort() | 要素をソートする |
| toString() | オブジェクトを文字列に変える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| concat() | ふたつの配列を連結し、新しい配列を返す |
| pop() | 配列から最後の要素を取り除き、その要素を返す |
| push() | 配列の最後へひとつ以上の要素を加え、配列の数を返す |
| shift() | 配列の先頭の要素を取り除き、配列数を返す |
| slice() | 配列の一部分を抜き出し、新しい配列を返す |
| splice() | 配列から要素を加えるか削除する |
| unshift() | 配列の先頭へひとつ以上の要素を加え、配列の数を返す |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

なし

## functionオブジェクト (JavaScript1.1)

関数の作成や操作を行うオブジェクト。

JavaScript1.1からnew演算子を使って新しい関数のオブジェクトを作成できるようになりました。argumentsはJavaScript1.0から存在しましたが、JavaScript1.1から正式にfunctionオブジェクトのプロパティになりました。

【オブジェクトの作成】

```
オブジェクト名= new Function ([要素1,要素2,...要素n], functionの働き)
```

【用法】

```
オブジェクト名.property
オブジェクト名.method(数値)
```

【プロパティ】

(JavaScript1.0)

未対応

(JavaScript1.1)

| | |
|---|---|
| arguments | ファンクションの要素(配列) |
| arguments.caller | ファンクションが何処から呼ばれたか |
| arguments.length | 渡されたファンクション配列の要素数 |
| length | 用意されているファンクションの要素数 |
| constructor | オブジェクトの作成元 |
| prototype | ファンクションのプロトタイプ |

(JavaScript1.2)

| | |
|---|---|
| arguments.callee | ファンクションの内容 |
| arity | 用意されているファンクションの要素数 |

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.3)

| | |
|---|---|
| call() | 異なるオブジェクトをコールする |
| apply() | 異なるオブジェクトをコールする（値を引き渡すことが可能） |
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

なし

## Objectオブジェクト

ユーザー独自の新しいオブジェクトを作成するオブジェクト。

【オブジェクトの作成】

```
オブジェクト名 = new Object()
```

【用法】

```
オブジェクト名.property
オブジェクト名.method()
```

【プロパティ】

(JavaScript1.0)

なし

(JavaScript1.1)

| | |
|---|---|
| constructor | オブジェクトの作成元 |
| prototype | オブジェクトのプロトタイプを作成 |

【メソッド】

(JavaScript1.0)

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |

(JavaScript1.1)

| | |
|---|---|
| valueOf() | オブジェクトの値を返す |

(JavaScript1.2)

| | |
|---|---|
| unwatch() | プロパティを監視を解除する |
| watch() | プロパティを監視する |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

(JavaScript1.1/JavaScript1.4)

| | |
|---|---|
| eval() | 文字列をJavaScriptのコードとして評価する |

※「eval()」メソッドは、元々javaScript1.0ではビルトイン関数でしたが、JavaScript1.1でメソッドとなり、JavaScript1.2では再びビルトイン関数になり、さらにJavaScript1.4からまたメソッドに戻りました。しかし、JavaScript1.5からは、再度ビルトイン関数に戻っています。

【イベントハンドラ】

なし

## Booleanオブジェクト (JavaScript1.1 ～)

真(true)の値と、偽(false)の値を作成するオブジェクト。

【オブジェクトの作成】

```
オブジェクト名 = new Boolean(値)
```

【用法】

```
オブジェクト名.property
オブジェクト名.method
```

【プロパティ】

(JavaScript1.0)

未対応

(JavaScript1.1)

| | |
|---|---|
| constructor | オブジェクトの作成元 |
| prototype | オブジェクトのプロトタイプの作成 |

【メソッド】

(JavaScript1.0)

未対応

(JavaScript1.1)

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

【イベントハンドラ】

なし

## Numberオブジェクト (JavaScript1.1～)

最大値などの属性を持った数値を作成するオブジェクト。

**【オブジェクトの作成】**

```
オブジェクト名 = new Number()
```

**【用法】**

```
オブジェクト名.property
オブジェクト名.method
```

**【プロパティ】**

(JavaScript1.0)

未対応

(JavaScript1.1)

| | |
|---|---|
| MAX_VALUE | 最大値を表す数値 |
| MIN_VALUE | 最小値を表す数値 |
| NaN | 値数が値ではないことを表す |
| NEGATIVE_INFINITY | マイナス方向の限界の値。越えると「overflow」を返す |
| POSITIVE_INFINITY | プラス方向の限界の値。越えると「overflow」を返す |
| constructor | オブジェクトの作成元 |
| prototype | オブジェクトのプロトタイプを作成 |

**【メソッド】**

(JavaScript1.0)

未対応

(JavaScript1.1)

| | |
|---|---|
| toString() | オブジェクトを文字列に変える |
| valueOf() | オブジェクトの値を返す |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

(JavaScript1.5)

| | |
|---|---|
| toExponential() | 指数関数的な記法の中で数を表す文字列を返す |
| toFixed() | 固定小数点式の記法の中で数を表す文字列を返す |
| toPrecision() | 指定された固定小数点式の記法の中で数を表す文字列を返す |

**【イベントハンドラ】**

なし

## RegularExpression (RegExp)オブジェクト (JavaScript1.2)

正規表現関連のオブジェクト。

**【オブジェクトの作成】**

```
正規表現 = /パターンの設定/[i|g|gi|m]
regexp = new RegExp("パターンの設定", ["i"|"g"|"gi"|"m"])
```

**【オプション(省略可能)】**

| | |
|---|---|
| i | 大文字･小文字を無視 |
| g | 完全一致 |
| gi | 大文字･小文字を無視して完全一致 |
| m | 複数の並びに一致 |

**【使用可能なメタ文字】**

| | |
|---|---|
| ¥ | 次に続く文字が通常に使われている文字の場合、そのまま解釈されずに特殊文字であることを示す。次に続く文字が通常特種文字として使われている文字の場合、特殊文字でないことを示す |
| ^ | 行頭に一致 |
| $ | 行末に一致 |
| * | 0回以上一致 |
| + | 1回以上一致({1,}と同じ) |
| ? | 0あるいは1回一致 |
| . | 任意の文字と一致 |
| (x) | 表現のグループ化。後方参照あり |
| (?:x) | 表現のグループ化。後方参照なし |
| x(?=y) | xがyに続く時のみ、xに一致 |
| x(?!y) | xがyに続かない時のみ、xに一致 |
| x¦y | xかyどちらかが一致 |
| {n} | n回一致(たとえば「/p{2}/」では、"Aple"は一致せず、"Apple"は完全一致、"Appple"は初めのふたつの"p"のみ一致) |
| {n,} | n回以上一致 |
| {n,m} | n回以上一致した所より、前m個が一致 |

リファレンス

| | |
|---|---|
| [xyz] | カッコ内の文字のどれでもひとつと一致(ハイフンの使用により、一連の文字を指定することができる。たとえば[abc]と[a-c]は同じ) |
| [^xyz] | カッコ内以外の文字と一致 |
| [¥b] | バックスペースと一致 |
| ¥b | スペースのように、単語との境界と一致 |
| ¥B | 単語との境界以外と一致 |
| ¥cX | 文字列の中の制御文字と一致(たとえば「/¥cM/」は、文字列内の"control-M"と一致) |
| ¥d | 数字と一致([0-9]と同じ。たとえば、"A1"では"1") |
| ¥D | 数字以外と一致([^0-9]と同じ。たとえば、"A1"では"A") |
| ¥f | フォームフィードと一致 |
| ¥n | ラインフィードと一致 |
| ¥r | キャリッジリターンと一致 |
| ¥s | スペースを含む余白やタブ、フォームフィードと一致([¥f¥n¥r¥t¥u00A0¥u2028¥u2029]と同じ) |
| ¥S | 余白以外の文字列と一致([^¥f¥n¥r¥t¥u00A0¥u2028¥u2029]と同じ) |
| ¥t | タブと一致 |
| ¥v | バーティカルタブと一致 |
| ¥w | 小文字・大文字も含むすべての英数字と一致([A-Za-z0-9_]と同じ。たとえば、"Apple"では"A"、"$9.18,"は"9") |
| ¥W | 小文字・大文字も含むすべての英数字以外と一致([^A-Za-z0-9_]と同じ。たとえば、"100%"では"%") |
| ¥n | n回目の一致 |
| ¥0 | "NUL"と一致 |
| ¥xhh | ふたつの16進数文字コードによって表された文字と一致。hhはふたつの16進数 |
| ¥uhhhh | Unicodeの文字コードによって表された文字と一致。hhhhは4つの16進数 |

**【プロパティ】**

(JavaScript1.2)

| | |
|---|---|
| global | 完全一致 |
| ignoreCase | 大文字・小文字を無視 |
| lastIndex | 次のパターンマッチを始める位置 |
| source | マッチさせるパターン |
| input ($_でも可) | 検索文字列の設定・変更 |
| multiline ($*でも可) | 改行コードを無視するかどうか。true(無視しない)かfalse(無視する)で設定 |
| lastMatch ($&でも可) | パターンがマッチした最後の文字 |
| lastParen ($+でも可) | パターンがマッチした最後のサブストリング |
| leftContext ($¥Qでも可) | パターンがマッチしたものの直前の文字 |
| rightContext ($'でも可) | パターンがマッチしたものの次の文字 |
| $1～$9 | パターンがマッチしたものの一部を蓄える(9個まで) |
| constructor | オブジェクトの作成元 |
| prototype | オブジェクトのプロトタイプを作成 |

**【メソッド】**

(JavaScript1.2)

| | |
|---|---|
| compile() | ループ文内で使うパターンを作成する |
| exec() | 指定したパターンと同じパターンを見つける |
| test() | 同じパターンがあるかどうかテストする |
| toString() | オブジェクトを文字列に変える |

(JavaScript1.3)

| | |
|---|---|
| toSource() | オブジェクトの値を文字列で返す |

**【イベントハンドラ】**

なし

# Top-Levelプロパティ

オブジェクトに関係なく使用できるプロパティ。

(JavaScript1.3)

| | |
|---|---|
| Infinity | 無限大を表している数値 |
| NaN | 数値でない値(Not A Number) |
| undefined | 不確定な値 |

# ビルトイン関数(top-level関数)

JavaScriptには、はじめから定義されている関数があり、それをビルトイン関数といいます。
ビルトイン関数は、オブジェクトに依存することなく、スクリプト内のどこからでも使用することができます。

(JavaScript1.0)

| | |
|---|---|
| parseInt() | 文字列を整数に変更する |
| parseFloat() | 文字列を浮動小数点に変更する |
| escape() | 文字をASCII形式に変換する(「unescape()」メソッドの逆) |
| unescape() | ASCII形式を文字に変換する(「escape()」メソッドの逆) |
| isNaN() | 値が数値かどうか判断する |
| eval() | 文字列をJavaScriptのコードとして評価する |

※「eval()」メソッドは、元々javaScript1.0ではビルトイン関数でしたが、JavaScript1.1でメソッドとなり、JavaScript1.2では再びビルトイン関数になり、さらにJavaScript1.4からまたメソッドに戻りました。しかし、JavaScript1.5からは、再度ビルトイン関数に戻っています。

(JavaScript1.1)

| | |
|---|---|
| taint() | データ参照を許可しないようにする |
| untaint() | 「taint()」メソッドを無効にする |

※「taint()」メソッド、および「untaint()」メソッドは、通常のブラウザでは機能しません。また、JavaScript1.2では削除されました。

(JavaScript1.2)

| | |
|---|---|
| Number() | オブジェクトを数に変換する |
| String() | オブジェクトを文字列に変換する |

(JavaScript1.3)

| | |
|---|---|
| isFinite() | 有限数かどうか判断する |

(JavaScript1.5)

| | |
|---|---|
| decodeURI | encodeURIや同等のルーチンでコード化されたURIを元に戻す |
| decodeURIComponent | encodeURIComponentや同等のルーチンでコード化されたURIコンポーネントを元に戻す |
| encodeURI | URIをコード化する |
| encodeURIComponent | URIコンポーネントをコード化する |

注意

## JavaScriptの2000年問題

JavaScript1.3では、西暦を4桁で表示する「getFullYear()」メソッドが追加されました。今後は、なるべくこのメソッドを使うことをお勧めします。しかしながら、現時点ではJavaScript1.3に未対応のブラウザが多く出回っているので、「getYear()」メソッドを使わなければいけない場合も多いでしょう。

「getYear()」メソッドは、基本的に西暦の下2桁を取得するメソッドなので、1999年は「99」、2000年になると「100」の値を返します。しかし、厳密に言うとJavaScript1.1では、1999年までは西暦の下2桁を取得し、2000年以降は4桁で取得するように仕様が変更されています。このため、Netscape Navigator 3.0などのJavaScript1.1対応ブラウザでは、1999年までは「99」と2桁の値を返しますが、2000年になると「getFullYear()」メソッドと同じように「2000」の値を返します。そして、さらに複雑なことに、「getFullYear()」メソッドがサポートされてからの「getYear()」メソッドは、再び2000年以降も西暦の下2桁を返すように戻されています。

つまり、「getYear()」メソッドは2000年になると、Netscape Navigator 2.Xでは「100」、Netscape Navigator 3.Xでは「2000」、Netscape Navigator 4.Xでは「100」を返すブラウザと「2000」を返すブラウザがあるのです。しかもこのうちNetscape Navigator 4.Xでは、どのバージョンが「100」を返し、どのバージョンが「2000」を返すかは、OSなどによって違いがあります。

このJavaScriptの2000年にまつわる問題を回避するには、例えば次のような方法があります。

```
<script type="text/javascript">
<!--
now = new Date();
    if ( now.getYear() >= 2000 ){ document.write(now.getYe
ar(),"年") }
    else { document.write(now.getYear()+1900,"年") }
//-->
</script>
```

こうしておけば、2000年以降を4桁で返す「getYear()」メソッドはそのまま表示され、それ以外の場合では、取得した値に「1900」を加えることによって、4桁の西暦に直して表示されます。

この方法は、「Dateオブジェクト」の「年・月・日・時・分・秒を表示する」(P.500)や「リアルタイムに年・月・日を表示する」(P.520)で使用しています。

# JavaScript付録コラム



**The revised edition of the detailed explanation dictionary about HyperText Markup Language, Cascading Style Sheets, and JavaScript.**

# DynamicHTMLとは

DynamicHTMLとは、大雑把にいえば、DOM(Document Object Model)をJavaScriptなどのスクリプトを使って操作し、ブラウザに動的な効果を与える技術ということになるでしょう。DOMとは、ブラウザに実装された機能やタグが持つ属性や要素などをオブジェクト化した技術でなので、DynamicHTMLに対応したブラウザは、それに対してスクリプトがアクセスする手段が用意されているというわけです。

たとえば本書では、サンプル「背景画像を変更する」(P.400)は「body」タグの「background」属性に、「altの値を返す」(P.469)は「img」タグの「alt」属性に、「画像の高さと幅を変更する」(P.474)は「img」タグの「width」属性と「height」属性に、「スタイルシートを移動する」(P.494)はスタイルシートの「left」属性に、それぞれJavaScriptを使ってアクセスし、値を取得したり、変更したりしており、これらはDynamicHTMLであるということができます。

DynamicHTMLでも特にスタイルシートは、オブジェクトの位置や可視属性などを細かく設定できるので、それをJavaScriptでコントロールすることにより、劇的な効果をブラウザで実現することが可能です。今では、単純なブラウジングの手助けだけでなく、DynamicHTMLで作ってブラウザで実行できるゲームなども、多数公開されています。

DOMは、Internet Explorerではバージョン4.0からサポートされています。また、Netscapeでも6.0からDOMがサポートされました。これにともない、Netscape 6.0では、JavaScript1.3以前のバージョンで言うところのナビゲータオブジェクトの部分もDOM化され、DOMの用法を用いてオブジェクトを操作することが可能になっています。

Internet Explorer 4.Xの頃は、W3CでDOMの仕様が規格化される前だったこともあり、DOMへアクセスし、これを操作してDynamicHTMLを実現するための方法は、独自色の濃いものでした。また、同じくDOMの規格以前のNetscape Navigator 4.Xも、コンテンツの位置決めにレイヤーかスタイルシートを、そしてオブジェクトの動きをJavaScriptで設定して、サイトに動的な効果を与えることをDynamicHTMLと言っていたため、Netscape NavigatorとInternet ExplorerのDynamicHTMLを実現する方法には、違いがありました。また、Netscape Navigator 4.XやInternet Explorer 4.Xの頃は、スタイルシートに関してもW3Cで仕様が規格化される以前だったこともあり、当時は両方のブラウザを対象にしたクロスブラウザのDynamicHTMLを作成するためには、スタイルシートの違いに関しても気を付ける必要がありました。

しかし、スタイルシートやDOMの仕様が確定した後に発表された、Netscape 6.0やInternet Explorer 6.0以降からは、スタイルシートの設定やDOMへのアクセスは、ほぼ同じ方法で行うことが可能になっています。

このように、DynamicHTMLは、ブラウザによって微妙な実装の違いがあり、これらのブラウザの実装の違いを吸収し、クロスブラウザで動くDynamicHTMLを作るには、ちょっとした工夫が必要です。DynamicHTMLを実現するためのDOMへのアクセス方法に関しては「DOMの値を取得する方法」(P.654)を、クロスブラウザDynamicHTMLを作る方法は「クロスブラウザDynamicHTMLを作るには」(P.656)を、それぞれ参考にしてください。

### ・関連ページ

**「The World Wide Web Consortium」**

(http://www.w3.org/)

**「Gecko DOM reference」**

(http://www.mozilla.org/docs/dom/domref/)

# DOMの値を取得する方法

DynamicHTMLのページを作成するためには、DOMにアクセスし、その値を取得する必要があります。DOMの各要素の値を取得するには、「getElementById(オブジェクト名)」メソッドを使うことによって可能です。この時のオブジェクトの名前の設定は、タグ内で「id」属性を使って行います。本来なら「name」属性も使えるはずなのですが、現状ではInternet ExplorerとNetscapeのどちらでもうまくいかない場合があるので、「id」属性を使った方がよいでしょう。

たとえば、次のようにすると、イメージオブジェクトに「IMG」という名前を付けたことになります。

```
<img id="IMG" src="image.jpg" width="100" height="100" alt="IMG">
```

このオブジェクトから、横幅の「width」属性の値を取得するには、次のように記述すればよいでしょう。

```
document.getElementById("IMG").width
```

この「getElementById()」メソッドは、Internet Explorer 5.0とNetscape 6.0から利用可能です。

しかしこれが、DOMの仕様が確定していなかったInternet Explorer 4.Xでは、「all(オブジェクト名)」メソッドを使ってDOMの値を取得することになります。たとえば、先ほどのイメージオブジェクトの場合は、次のようになります。

```
document.all("IMG").width
```

「all()」メソッドは、Internet Explorer 4.X以降の5.0や6.0などでも、引き続きサポートされています。

ちなみに、「id」属性を使ってオブジェクト名を設定した場合、Netscape Navigator 4.Xでは、スタイルシート以外で、「document.IMG.width」といったようなJavaScriptの用法を使用したオブジェクトの値の取得はできません。DOMの用法とJavaScriptの用法を混在して使用する必要がある場合は、次のように「id」属性と「name」属性の両方を使って、オブジェクト名を設定すればよいでしょう。

```
<img id="IMG" name="IMG" src="image.jpg" width="100" height="100" alt
="IMG">
```

なお、スタイルシートに関しては、Netscape Navigator 4.Xで「id」属性を使ってオブジェクト名を設定しても、値を取得することができます。

DynamicHTMLで動的なページを作る場合、見栄えを細かく設定することが可能な

スタイルシートを使って実現するのは、大変に有効な手段といえます。スタイルシートの場合も、Internet Explorer 5.Xや6.X、Netscape 6.Xなどでは「getElementById()」メソッドを使って、Internet Explorer4.0以降のブラウザは「all()」メソッドを使って値を取得することが可能です。

また、Netscape Navigator 4.Xでも、DynamicHTMLがサポートされていることになっています。しかしその範囲は、現在ではNetscape Navigator 4.X独自の仕様になってしまったレイヤーや、実装に多くの問題があるスタイルシートに限られています。そして、Netscape Navigator 4.Xでは、「layers[オブジェクト名]」を使って、レイヤーやスタイルシートの値を取得することになります。

たとえば、次のようなスタイルシートから、スタイルシートの画面左端からの位置の値を取得する場合、それぞれのブラウザでは、次のようにします。

**【スタイルシート】**

```
<div id="STY1" style="position:absolute; left:100px; top:50px; width
:200px; height:150px;background:Tan">
<b>STY1...</b>
</div>
```

**【Internet Explorer 5.X ～ 6.X、Netscape 6.X】**

```
document.getElementById("STY1").style.left
```

**【Internet Explorer 4.X ～ 6.X】**

```
document.all("STY1").style.left
```

**【Netscape Navigator 4.X】**

```
document.layers["STY1"].left
```

本書では、「getElementById()」メソッドや「all()」メソッドを使ってDOMの値を取り出したり、DOMに値を設定し直すことによってページの内容を変化させるサンプルをいくつか紹介していますので、合わせて参考にしてください。

また、Netscape Navigator 4.Xにおけるスタイルシートの実装上の問題に関しては、付録コラム「Netscape Navigator 4.Xでのスタイルシートの問題点」(P.660)を参考にしてください。

## クロスブラウザDynamicHTMLを作るには

クロスブラウザのDynamicHTMLを作る場合、ブラウザの種類やバージョンを考慮し、それぞれに合わせた用法を用いたスクリプトを実行するようにする必要があります。ブラウザの種類やバージョン、あるいは、そのブラウザがどのような機能をサポートしているかを判断するには、色々な方法があります。ここでは、if文の条件式にオブジェクトを設定することによって、そのオブジェクトをサポートするブラウザでのみスクリプトを実行させる方法について解説します。この方法は、本書でもクロスブラウザDynamicHTMLを実現するために利用していますので、そのサンプルも見ながら解説していきたいと思います。

それでは、if文に設定する条件式と、スクリプトが実行されるブラウザについて見ていきましょう。

次のように、if文の条件式に「document.getElementById」を設定した時、if文内で設定したスクリプトは、「getElementById()」メソッドをサポートしたブラウザ、つまりInternet Explorer 5.0やNetscape 6.X以降のバージョンのブラウザでのみ実行されます。

```
if(document.getElementById){ Internet Explorer 5.X～6.X、Netscape 6.X で実行されるスクリプト }
```

また、次のようにif文の条件式に「document.all」を設定した時、if文内で設定したスクリプトは、「all()」メソッドをサポートしたブラウザ、つまりInternet Explorer 4.0以降のバージョンのブラウザでのみ実行されます。

```
if(document.all){ Internet Explorer 4.X～6.X以上で実行されるスクリプト }
```

さらに、次のようにif文の条件式に「document.layers」を設定した時、if文内で設定したスクリプトは、「layers[]」をサポートしたブラウザ、つまりNetscape Navigator 4.0のブラウザでのみ実行されます。

```
if(document.layers){ Netscape Navigator 4.Xで実行されるスクリプト }
```

それでは、実際にこれらをどのように使っているか、「クリックした位置へスタイルシートを移動する」(P.497)を例にとって、見てみましょう。

このサンプルでは、Internet Explorer 5.X ～ 6.X用、Internet Explorer 4.0用、Netscape 6.X用、Netscape Navigator 4.0用にそれぞれスクリプトを用意しています。

サンプルからブラウザの判断部分のみを取り出すと、次のようになります。

```
if(navigator.appName.charAt(0)=="M"){
        if (navigator.appVersion.indexOf("MSIE 5") != -1 || navigator.appVersion.indexOf("MSIE 6") != -1){
```

付録コラム

```
            if(document.getElementById){ Internet Explorer 5.0、Inte
rnet Explorer 6.0用のスクリプト }
        }
        else{
            if(document.all){ Internet Explorer 4.0用のスクリプト }
        }
    }
    if(navigator.appName.charAt(0)=="N"){
        if(document.getElementById){ Netscape 6.X用のスクリプト }
        if(document.layers){ Netscape Navigator 4.0用のスクリプト }
    }
```

ここでは、まず始めに「if(navigator.appName.charAt(0)=="M")」として、ブラウザがInternet Explorerである時の処理を設定しています。そして次には、Internet Explorer 5.0と6.0用のスクリプトとInternet Explorer 4.0用のスクリプトのそれぞれを用意しているので、これらのブラウザを判断する必要があります。

そこで、「if (navigator.appVersion.indexOf("MSIE 5") != -1 ¦¦ navigator.appVersion.indexOf("MSIE 6") != -1)」としてブラウザのバージョンを判断することによって、Internet Explorer 5.0か6.0であることを判断します。さらに「if(document.getElementById)」とすることによって、Internet Explorer 5.0か6.0で、かつ「getElementById()」メソッドに対応したブラウザのみ「getElementById()」メソッドを使ったスクリプトを実行するようにしています。

Internet Explorer 5.0や6.0は「getElementById()」メソッドがサポートされているので、この処理は一見不要のように思えます。しかし、ブラウザの中には、ユーザーエージェントをユーザー自身が設定できるものもあり、またそのブラウザがJavaScriptをサポートしていたとしても、「getElementById()」メソッドをサポートしているとは限りません。このようなブラウザの対策のためにも、「if(document.getElementById)」の処理は、必要でしょう。

さらに、Internet Explorer 4.0用のスクリプトは、次の「else」の処理内の、「if(document.all)」内に設定しています。「else」内の処理はInternet Explorer 3.0でも実行されますが、「if(document.all)」の処理を入れているので、Internet Explorer 3.0ではif文内で設定したサポートされていない「all()」メソッドを使ったスクリプトが実行されることはありません。

なお、Internet Explorer 5.0や6.0でも、「all()」メソッドがサポートされています。そのため、ブラウザのバージョン判断の処理をしないと、「getElementById()」メソッドと「all()」メソッドの両方の処理が実行されてしまうので、ブラウザのバージョン判断の処理は、必ず行うようにしてください。

Netscape用の処理は、「if(navigator.appName.charAt(0)=="N")」として、ブラウザがNetscapeであることを判断した後に設定しています。

「getElementById()」メソッドに対応したNetscape 6.X用のスクリプトは「if(document.getElementById){」として、「layers[]」に対応したNetscape Navigator 4.0用のスクリプトは「if(document.layers)」として、ブラウザが実行できるスクリプトを判断します。その後、Netscape 6.X用には「getElementById()」メソッドを、Netscape Navigator 4.0用には「layers[]」を使ったスクリプトを、それぞれ設定しています。

Netscape 6.Xは「layers[]」を、反対にNetscape Navigator 4.0は「getElementById()」メソッドをサポートしていません。このためNetscapeでは、Internet Explorerのような細かなバージョン判断の処理はいりません。

以上のようなブラウザの判断方法は、あくまでもほんの一例です。ブラウザやブラウザのバージョン判断の内容は、スクリプトがどのブラウザのどのバージョンで実行することができるか、あるいは実行したいかによって変わってきます。また、if文の条件式も「これが唯一の正解」といったものではありません。多くのスクリプトを見て、色々と試してみることをお勧めします。

ところで、Internet Explorer 5.0と6.0は「all()」メソッドもサポートしているので、「クリックした位置へスタイルシートを移動する」(P.497)のサンプルから、次のようにInternet Explorer用に設定したスクリプトから「getElementById()」メソッドを使って設定した部分をはぶいてもかまいません。

```
<html><head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title></title>
<script type="text/javascript">
<!--
function eve(e) {
    if(navigator.appName.charAt(0)=="M"){
            if(document.all){
                document.all("STY").style.left=window.event.offsetX;
                document.all("STY").style.top=window.event.offsetY;
        }
    }
    if(navigator.appName.charAt(0)=="N"){
        if(document.getElementById){
            document.getElementById("STY").style.left=e.pageX;
            document.getElementById("STY").style.top=e.pageY;
        }
        if(document.layers){
            document.layers["STY"].left=e.pageX;
            document.layers["STY"].top=e.pageY;
```

付録コラム

```
        }
    }
}
document.onmousedown = eve;
//-->
</script>
<style type="text/css">
<!--
body { background-color: #ffffff; }
-->
</style>
</head>
<body>
*クリックした位置へスタイルシートを移動する
<div id="STY" style="position:absolute; width:100px; height:100px; 
 left:50px; top:50px">
<img src="image.jpg" alt="image.jpg" width="100" height="100">
</div>
</body></html>
```

これは、これ以外の「スタイルシートのクリップサイズを変更する」(P.488)や「スタイルシートの表示・非表示を切り替える」(P.491)、「スタイルシートを移動する」(P.494)などの、Internet Explorer用のスクリプトの中で「all()」メソッドと「getElementById()」メソッドを使用しているサンプルでも同じです。

しかし本書では、Internet Explorer 5.0以降のブラウザで「getElementById()」メソッドが使えることを明確に示したかったので、あえて「all()」メソッドと「getElementById()」メソッドの処理を分けて設定しています。

さて、このif文の条件式にオブジェクトを設定することによって、そのオブジェクトをサポートするブラウザでのみスクリプトを実行させるテクニックは、何もDynamicHTMLの時のみに有効な手段というわけではありません。たとえば、次のようにif文の条件式にImageオブジェクトを設定すると、if文内に設定したスクリプトは、Netscape Navigator 3.0以降、あるいはInternet Explorer 3.0以降といったJavaScript1.1に対応したブラウザでのみ、実行されることになります。

```
if(document.images){ JavaScript1.1に対応したブラウザで実行されるスクリプト }
```

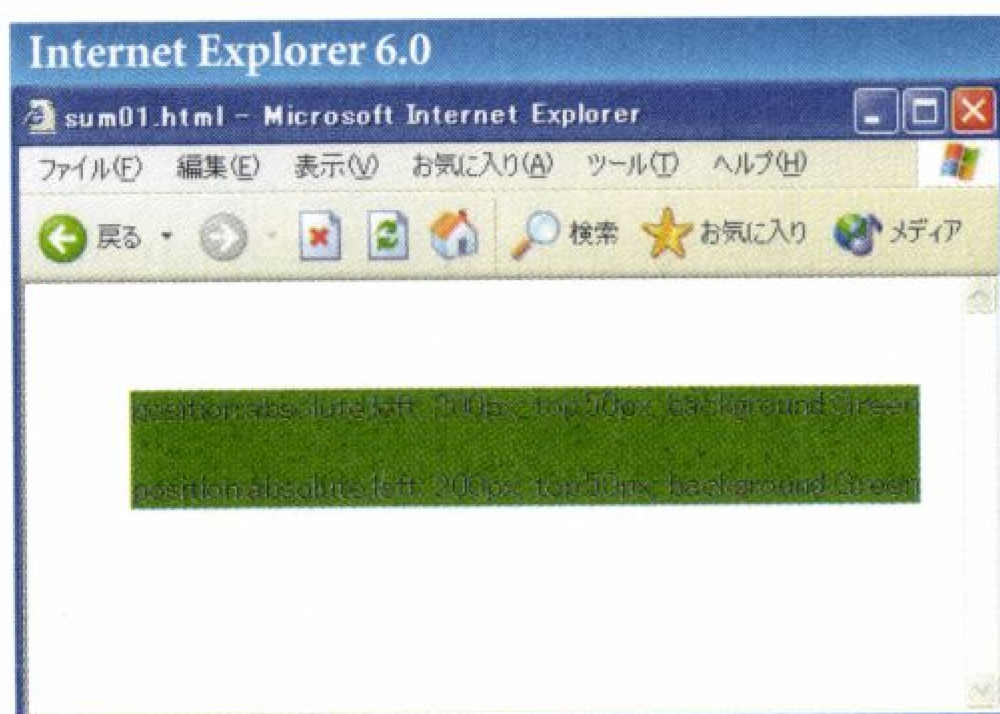
HTML4.0からは、「script」タグの「language」属性は、不適切とされています。これにより、付録コラム「JavaScriptのバージョン記述によるエラー回避」(P.667)で紹介したような、「language」属性にJavaScriptのバージョンを記述することによるエラー回避よりも、今後は、ここで紹介したif文の条件式に実行するスクリプトがサポートしているオブジェクトを設定することによって行うエラー回避が、主流になってくることでしょう。

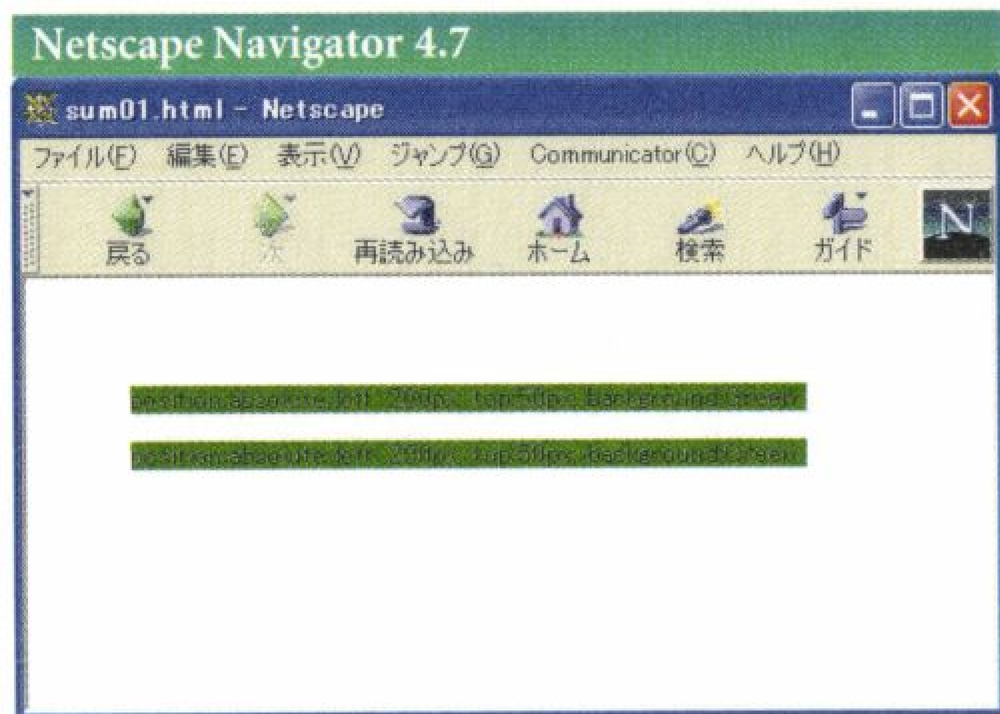
# Netscape Navigator 4.Xでのスタイルシートの問題点

Netscapeは、Netscape Navigator 4.0からスタイルシートがサポートされました。しかし、Netscape Navigator 4.Xのスタイルシートの実装にはいくつかの問題があり、背景色や背景画像の表示がうまく行えないので、注意してください。

たとえば、次のスタイルシートをInternet ExplorerやNetscape 6.0とNetscape Navigator 4.Xで実行した場合、

```
<div id="STY1" style="position:absolute; top:50px; left:50px; backgr
ound:Green">
<p>
position:absolute;left: 200px; top:50px; background:Green
</p>
<p>
position:absolute;left: 200px; top:50px; background:Green
</p>
</div>
```



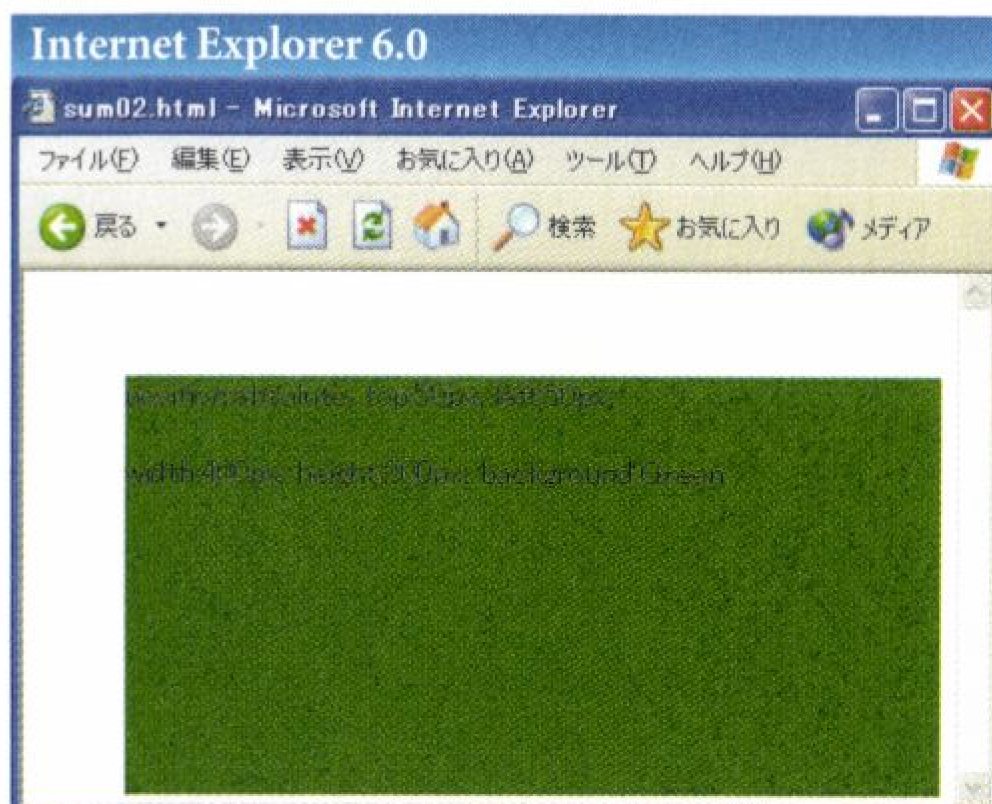



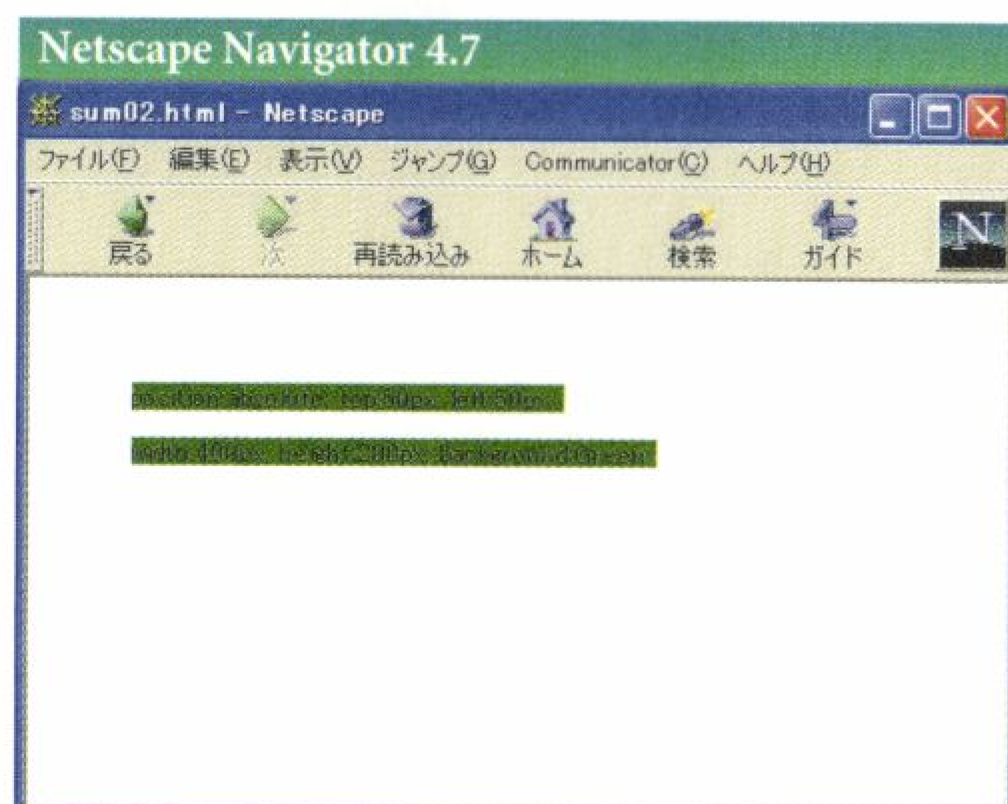

Internet ExplorerやNetscape 6.0では、行と行の間もスタイルシートで設定した背景色になりますが、Netscape Navigator 4.Xでは、文字の後ろしか背景色が変わらず、まだらになってしまうことがわかると思います。

さらにこの問題は、次のサンプルのように、「width:」と「height:」を使用してスタイルシートのサイズを明示的に設定しても発生します。

```
<div id="STY1" style="position:absolute; top:50px; left:50px; width
:400px; height:200px; background:Green">
<p>
position:absolute; top:50px; left:50px;
</p>
<p>
```

```
width:400px; height:200px; background:Green
</p>
</div>
```




Internet ExplorerやNetscape 6.0では設定通りに幅100ピクセルで高さ200ピクセルのスタイルシート全体の背景色が表示されますが、Netscape Navigator 4.Xでは、背景色が変るのは文字がある部分のみです。

また、次のサンプルは、本文中のサンプル「スタイルシートの情報を取得する」(P.482)に<meta>を設定したものです。しかし、このように文字だけのスタイルシートの場合、Netscape Navigator 4.Xでは、スタイルシートの部分が表示されない場合があります。

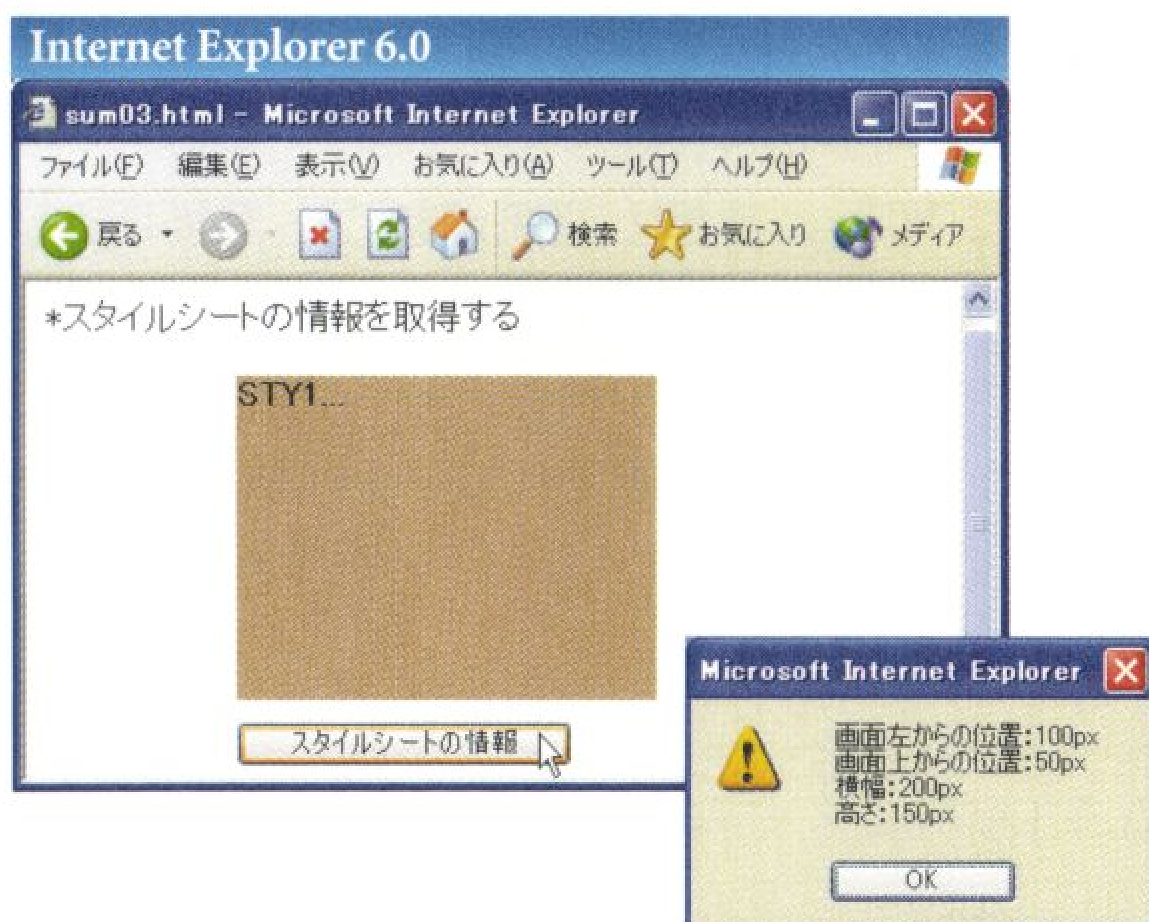
```
<html>
<head>
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<title>sum03.html</title>

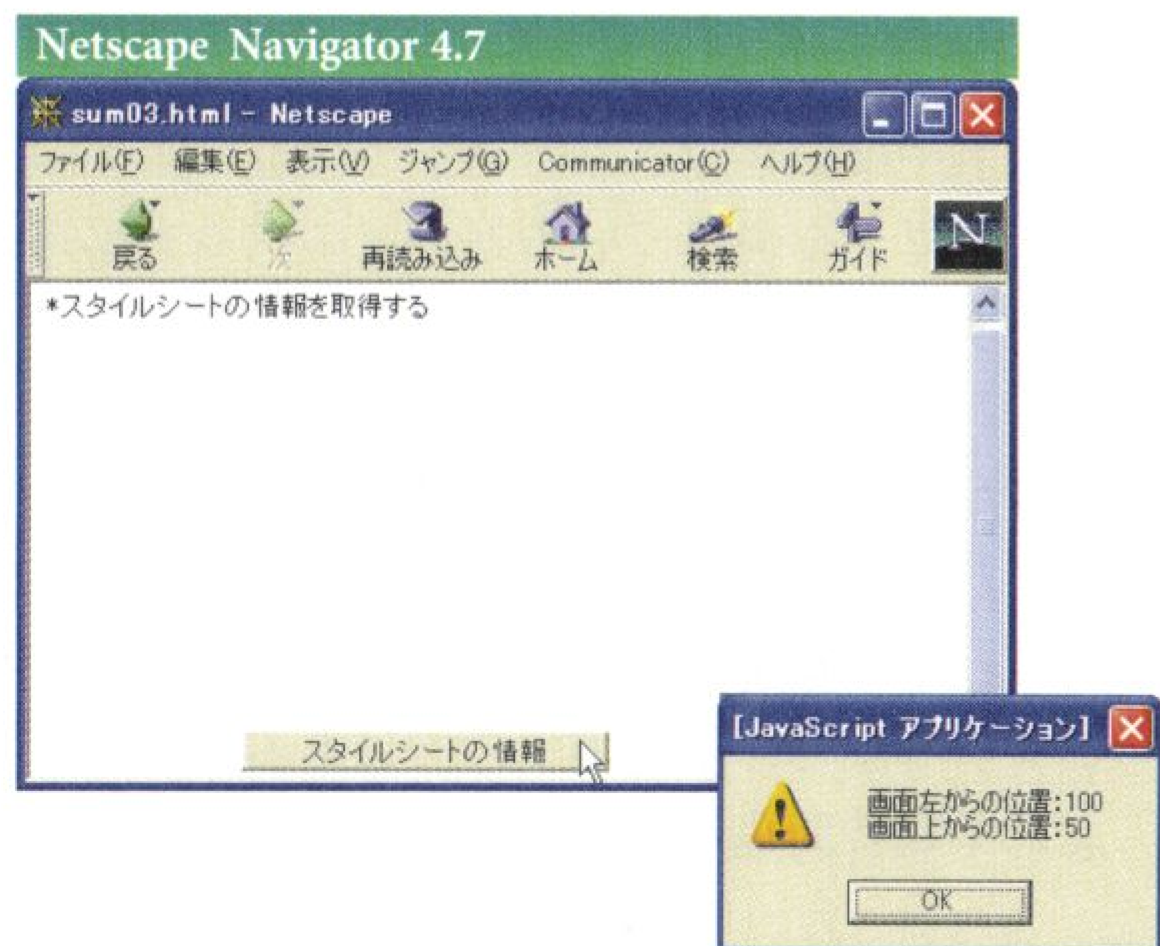
<script type="text/javascript">
<!--
function StyValue(){
    if(navigator.appName.charAt(0)=="M"){
        if(document.all){
            alert("画面左からの位置："+document.all("STY1").style.left+
"¥n"+
                "画面上からの位置："+document.all("STY1").style.top+"
¥n"+
                "横幅："+document.all("STY1").style.width+"¥n"+
                "高さ："+document.all("STY1").style.height)
        }
```

```
    }
    if(navigator.appName.charAt(0)=="N"){
        if(document.getElementById){
            alert("画面左からの位置："+document.getElementById("STY1").
style.left+"¥n"+
                "画面上からの位置："+document.getElementById("STY1").
style.top+"¥n"+
                "横幅："+document.getElementById("STY1").style.wid
th+"¥n"+
                "高さ："+document.getElementById("STY1").style.hei
ght)
        }
        if(document.layers){
            alert("画面左からの位置："+document.layers["STY1"].left+"¥n"+
                "画面上からの位置："+document.layers["STY1"].top)
        }
    }
}
//-->
</script>

</head>
<body bgcolor="#FFFFFF">
*スタイルシートの情報を取得する
<div id="STY1" style="position:absolute; left:100px; top:50px;
 width:200px; height:150px;background:Tan">
<b>STY1...</b>
</div>
<div id="STY2" style="position:absolute; left:100px; top:210px">
<form>
    <input type="button" value="スタイルシートの情報" onClick="StyValue()">
</form>
</div>
</body>
</html>
```

付録コラム




これらの問題を回避するためは、「スタイルシートと同じ大きさの画像をスタイルシートに設定する」「ブラウザを判断して、Netscape Navigator 4.Xの場合はレイヤーで、それ以外の場合はスタイルシートでページをレイアウトする」などの方法があります。

※これらのサンプルのソースコードは、サポートページで配布しているサンプルファイルの「js」−「column」−「nn_css」の中に「sum01.html」〜「sum03.html」として収録してあります。

# JavaScriptのありがちなミス

JavaScriptのソース上に問題があった場合、問題の種類と問題が起こった場所が、Internet Explorerの場合はエラーウィンドウ、Netscapeの場合はJavaScriptコンソールに表示されます。JavaScriptでスクリプトを組む時には、この情報を元にデバッグしていくことになります。しかしながら、このウィンドウで指摘されている問題があった場所は、実際にはこの場所と別の場所に不具合があったためにエラーが引き起こされる場合もありますので、気を付けてください。

著者の経験上でスクリプト作成時に起こしやすいミスをまとめてみました。ブラウザでチェックする前に、以下の項目をもう1度確認してみてください。

## 記述上のミス

**・名称のスペルミス**

関数名や変数名は、ユーザーが規定の範囲内で自由に定義することができますが、スクリプトを書いている途中でスペルを間違えたり、抜かしてしまうことがよくあります。また、大文字・小文字も区別するので、その点にも注意が必要です。

**・オブジェクトの階層のミス**

ナビゲータオブジェクトには、オブジェクトに階層関係があり、それらを使用する時は、その階層関係を間違えずに記述して使用する必要があります。特にTextオブジェクトなど、階層の下の方にあるオブジェクトでは、途中のオブジェクトの記述忘れや順番間違いに気を付けてください。

**・オブジェクト名・関数名・変数名の付け方の注意**

オブジェクト名・関数名・変数名には、自分が見てわかりやすい名前を付けるようにしましょう。また、これらに同じ名前を使ってしまうことが、意外と多いようです。スクリプトが動かない場合は、この点にも注意してチェックするとよいでしょう。

**・関数名・変数名の取り違え**

長いスクリプトを書いている時に、特に変数名などは、値の代入を繰り返していくうちに、どの変数にどの値が代入されたかが、こんがらがってしまうことがあります。これらの名称は、自分自身が見てわかりやすい名称を設定することをお勧めします。また、複雑な処理をしているところでは、どういう処理をしているかをコメントに書いておくのもよいでしょう。

**・関数名のダブり**

ひとつのHTMLファイル内に同名の複数の関数がある場合、1番最後に記述された関数のみ動作し、他の関数は無効になってしまいます。

・"{~}"などのカッコや"~"の閉じ忘れ

処理文が長くなったり、何重にもネスト(入れ子)させていると、処理文を閉じる「}」を付け忘れたり、間違った所に付けてしまうことがよくあります。各階層ごとに段落を下げたり、コメントを付けるなどの工夫をすることをお勧めします。また、文字列を閉じる「"」なども、文字列が長くなったり、変数と一緒に使ったりしていると、うっかり付け忘れてしまいがちです。

・"と'のネスト(入れ子)の間違い

「"~"」で括られた文字列内では、「"」を使わず「'」を使用します。うっかり「"」を使ってしまうと、JavaScriptはそこで文字列が終わったと判断し、エラーになってしまいます。特にイベントハンドラ内で、文字列などを設定する時に間違えてしまいがちですので、注意が必要です。

・「=」ではなく「==」

JavaScriptでは、左右の値を評価して値が同じであることをあらわす時は、「値A = 値B」ではなく「値A == 値B」と記述します。

## 文字コード上の問題

・「表示」「予定」などの文字化けする文字によるエラー

「document.write()」で文字を書き出した時に文字化けするだけでなく、文字の組み合わせによってはエラーになってしまう場合があります。

・スペースなのに「不正な"@"があります」と言われた時

スペースの表現に半角スペースや全角スペース、タブを混在して使用している時、何も問題がないはずなのに、文字コードの関係で「不正な@があります(英文)」というエラーが出る時があります。この問題を回避するため、スペースを空ける時には、半角英数のスペースに統一して使用することをお勧めします。また、不要なスペースは極力削除しておくこともお勧めします。

## レイアウト上の問題

・大きめの画像の後にJavaScriptを記述した時

Netscape Navigator 2.Xでは、大きな画像ファイルやカウンターなどの表示に時間がかかるものがあると、それ以降のJavaScriptが動かない場合があります。この問題は、画像ファイルの前にスクリプトを記述したり、「width」属性や「height」属性を指定することによって、ほぼ解決されます。

・テーブル内やテーブル後にJavaScriptを記述した時

テーブル内、あるいはテーブルのすぐ後にJavaScriptを記述した時も、スクリプトが動かない場合があります。この問題は、Netscape Navigator 2.Xや3.Xで見られ、テーブルのサイズに余裕を持たせたり、テーブルより前にスクリプトを記述することによって、ほぼ解決されます。

### ・アニメーションGIFと同時に使用する時の注意

Netscape Navigator 2.Xで、アニメーションGIFとJavaScriptを同時に使用しているページを見た場合、ページの読み込みが途中で止まったり、ブラウザが不安定になってシステムエラーなどを引き起こす場合があります。この問題は、Netscape Navigator 3.0では解決していますが、それ以前のブラウザを対象にしたページでは、アニメーションGIFとJavaScriptの同時使用は避けた方がよいでしょう。

## 「window.open()」を使用する時の注意

### ・開いたウィンドウに文字が記述されない・最後の1行が表示されない

「window.open()」で開いたウィンドウに「document.write()」で文字を書き出した時、「document.close()」でドキュメントストリームを明示的に閉じていないと、全く表示されなかったり、最後の1行が表示されなかったりする場合があります。

### ・「window.open()」で開いたウィンドウ内の画像はフルパスで

「window.open()」で開いたウィンドウに画像ファイルを表示させる時、Netscape Navigator 2.Xなどの一部のブラウザでは、URLをフルパスで指定していないと画像が表示されない場合があります。

## デバック時のTips

### ・「setTimeout()」メソッドは時間設定を大きめにしてテストする

リアルタイムの時間表示やアニメーションするJavaScriptのような、一定時間に処理を繰り返すスクリプト中にミスがあると、エラーを表示するウィンドウが繰り返し出て、どうしようもない状態になる時があります。このようなスクリプトをデバックする時には、繰り返し時間の設定を長めにして、問題がなくなってから設定時間を短くすることをお勧めします。

### ・ブラウザは1度終了させてから

キャッシュ機能が働くため、ブラウザの[更新(Reload)]ボタンを押しただけでは、HTMLファイルが更新されない場合があります。もし記述ミスや構文ミスが何もないのにまだエラーが出る時には、1度ブラウザを終了させてからファイルを読み込み直してみてください。

# JavaScriptのバージョン記述によるエラー回避

<script>内の「language」属性の指定で、「language=JavaScript1.1」や「language=JavaScript1.2」といった具合にJavaScriptのバージョンを記述をすれば、ページを読み込む時に、そのバージョンのブラウザでは未対応のJavaScriptが記述してあってもエラーになることはありません。

しかしこのままでは、イベントハンドラでイベントを発生させた時などにエラーを起こしてしまう場合があります。このような問題を回避するためのひとつの方法として、ダミーのスクリプトを設定するという方法があります。

たとえば、「Imageオブジェクト」の「アニメーションにスタートボタンとストップボタンを付ける」(P.458)の場合、このままImageオブジェクトに未対応のNetscape Navigator 2.Xのブラウザでこのページを見ると、ページの読み込み時には何も起こらないのですが、[スタート]あるいは[ストップ]ボタンを押した途端、エラー表示のウィンドウが開いてしまいます。

このような場合、次のように、「language」属性を使い、さらに画像ファイルをアニメーションさせるスクリプトの前に、何もしないダミーのスクリプトを記述すると、Netscape Navigator 2.Xでは「language」属性がJavaScript1.1のスクリプトには反応しないので、ダミーのスクリプトが評価され、エラー表示のウィンドウは表示されません。

```
<script language="JavaScript">
<!--
function anime_2(){ }
function stop(){ }
//-->
</script>
<script language="JavaScript1.1">
<!--
var TimeSet1 = 500 ;
var ImageSetA = 1 ;
  ⁝
```

また、Netscape Navigator 3.0以上のブラウザでは両方のスクリプトが認識されますが、同じ関数名があった場合、より後に記述された関数が評価されるので、正常にスクリプトが実行されます。

もちろん、ダミーの部分にNetscape Navigator 2.X用の処理を設定してもかまいません。本書では、この他にも、ブラウザの種類やバージョンによってページを振り分けるなどの方法を紹介していますので、そちらも参考にしてください。

# エラーウィンドウについて

Netscape Navigator 4.06からJavaScript1.3がサポートされたのとともに、今まで問題があるJavaScriptのコードを含んだHTMLファイルをブラウザで読み込んだ時に表示されていた、JavaScript Errorのウィンドウが表示されなくなりました。

この対応は、ネットサーフィンをしている最中に、ユーザーは何も悪くないのに突然エラーウィンドウが表示されることがなくなったということなので、ユーザに対しては適切な対応といえます。

なお、JavaScriptの動作チェックをする場合などは、ロケーションバーに「javascript:」と打ち込んで開くコンソールウィンドウに、エラーが発生した時のエラーの発生場所や内容が表示されるので、これでデバックを行うことになります。また、「Netscape Preferences」ファイルに以下の1行を追加しておくと、JavaScriptエラーが発生した時に、自動的にエラー内容を表示したコンソールウィンドウが開くようになります。

```
user_pref("javascript.console.open_on_error", true);
```

JavaScriptのデバックを行うことが多く、コンソールウィンドウは常に表示されるようになっていて欲しいのであれば、これを設定しておくのもよいでしょう。また、「Netscape Preferences」の設定でコンソールウィンドウを開くように設定した後も、以下のように「false」を設定することにより、コンソールウィンドウを開かないようにすることができます。

```
user_pref("javascript.console.open_on_error", false);
```

なお、Netscape 7.X以降のMozilla、Firefoxを含めた、Mozilla系のブラウザの場合は、[タスク] − [ツール] − [Web開発] − [JavaScriptコンソール] といったように、メニューからの選択でも、コンソールウィンドウを開くことができます。

Windows版のInternet Explorerの場合は、エラーがあると、ウィンドウの左下に[!]マークが表示されるとともに、エラーウィンドウが開きます。エラーウィンドウの[詳細の表示]ボタンをクリックするとエラーの詳細が表示され、ボタンが[詳細を隠す]に変わります。その[詳細を隠す]ボタンをクリックすると、エラーの詳細が隠れます。エラーウィンドウ内の「ページにエラーがある時は、このエラーメッセージをいつも表示する」のチェックボックスのチェックを外すと、ページにエラーがあってもエラーウィンドウが開かなくなります。しかし、この場合でも、ウィンドウの左下に[!]マークが表示されるので、そこをダブルクリックすることによって、エラーウィンドウを表示することが可能です。

また、Macintosh版のInternet Explorerの場合は、「初期設定」の「Webコンテンツ」の項目から、「アクティブコンテンツ」にある「スクリプトエラーの警告を表示」のチェックボックスのチェックを外すと、エラーウィンドウの表示を停止することができます。

# JavaScript1.2で追加・変更された用法

ここでは、JavaScript1.2で追加・変更された用法のうち、本文中で解説しきれなかった事柄について、まとめています。

## 追加・変更されたArrayオブジェクトの用法

次の用法で配列を作成できるようになりました。

**【用法】**

```
配列名 = [element0, element1, ..., elementn]
```

サンプルは、「Arrayオブジェクト」の「配列の要素を文字列にして書き出す」(P.564)のサンプルを、この用法を使って書き直したものです。

```
<script language="JavaScript1.2">
<!--
hairetu= ["0番目","1番目","2番目","3番目"];
document.write(hairetu.join("/"));
//-->
</script>
```

また、次のサンプルの場合、JavaScript1.2以前では「undefined」となりますが、JavaScript1.2では「1」と値を返すようになりました。

```
<script language="JavaScript1.2">
<!--
a = new Array(1)
document.write(a[0]);
//-->
</script>
```

## 変更されたtoStringの値

オブジェクトを文字列に変換する「toString()」メソッドをJavaScript1.2で使用すると、JavaScript1.2で追加されたオブジェクトや配列の形式で書き出されます。

たとえば、下のサンプルの場合、書き出される文字列は、「[object Object]」とならずに「{color:red, size:2.2}」となります。

```
<script language="JavaScript1.2">
<!--
mystring = new Object();
mystring.color = "red";
```

```
mystring.size = 2.2;
document.write(mystring.toString());
//-->
</script>
```

## 変更されたNumberオブジェクトの用法

下のサンプルのように「Number()」メソッドで数値以外の値を設定すると、Netscape Navigator 4.0以前のブラウザではエラーになりますが、JavaScript1.2に対応したNetscape Navigator 4.0からは、「NaN」の値を返すようになりました。

```
<script language="JavaScript">
<!--
x = Number("one")
document.write(x);
//-->
</script>
```

## 変更されたstringオブジェクトの用法

文字列を分割する「split()」メソッドをJavaScript1.2で使用すると、「*文字列*.split(" ")」としてスペースで分割するように指定しても、タブ・改行などの特殊記号や、複数のスペースなどが認識されるようになりました。

たとえば、下のサンプルをJavaScript1.2で実行した場合、結果は「["Welcome", "To", "My", "Home", "Page"]」となり、それ以前のバージョンのJavaScriptでは「Welcome,,To, My,,,,Home, Page」となります。

```
mozi1="Welcome  To ¥nMy    Home ¥nPage"
a=mozi1.split(" ")
document.write(a)
```

文字列の途中の文字を抜き出す「substring()」メソッドは、JavaScript1.2で使用した時は設定された値を厳密に判断するようになりました。

たとえば、次のサンプルをJavaScript1.2で実行した場合、抜き出す先頭の文字の値を0以下の数値に設定したり、抜き出す先頭の文字より抜き出す終了位置の文字を大きな数値に設定した時は、エラーになります。

```
var mozi1 = "Welcome To My Home Page";
var mozi2 = "ようこそ私のホームページへ";
    document.write(mozi1.substring(-2,12));
    document.write("<br>");
    document.write(mozi2.substring(12,8));
```

## RegularExpression（正規表現）について

JavaScript1.2から、一定パターンの文字・キャラクターを設定し、それを使って検索や置き換えができるようになりました。

パターンの設定は、設定する文字やキャラクターを「/」で括ることによって行います。パターンの認識方法は、たとえば「/abc/」と設定した場合は、完全に同じ「abc」という文字列でなければパターンが合っていると認識されません。また「/ab*c/」と設定した場合では、「abc」だけでなく"b"の後に複数の"b"がある"abbbc"のような時もパターンが合っていると認識されます。

パターンの設定方法は、この他にもたくさんあります。

RegularExpression（正規表現）関連では、stringオブジェクトにメソッドが追加されるとともに、パターンの設定を取り扱うRegularExpressionオブジェクトが追加されています。

基本的なRegularExpressionの用法は、以下の通りです。

また、正規表現自体の用法に関しては、多くの有用な書籍が刊行されていますので、参考にするといいでしょう。

このサンプルでは、stringオブジェクトの「match(*パターンの設定*)」メソッドを使って、一定のパターンの文字と合った文字を検索しています。

```
<script language="JavaScript1.2">
s1 = "NetscapeNavgator4.0";
foun = s1.match(/Nav/);
document.write(foun);
</script>
```

次のサンプルでは、stringオブジェクトの「replace(*パターンの設定*,"*入れ替える文字列*")」メソッドを使って、指定した文字列を別の文字列に入れ替えています。

```
<script language="JavaScript1.2">
s2 = "NetscapeNavgator4.0";
repl = s2.replace(/Navgator/,"Communicat");
document.write(repl);
</script>
```

次のサンプルでは、stringオブジェクトの「split(*パターンの設定*)」を使って、パターンが合った部分で文字列を分割しています。

```
<script language="JavaScript1.2">
s3 = "NetscapeNavgator4.0";
spli = s3.split(/a/);
document.write(spli);
</script>
```

次のサンプルでは、「new RegExp("*パターンの設定*")」メソッドを使ってRegular Expressionオブジェクトを作成し、「exec(*検索する文字列*)」メソッドを使って一定のパターンの文字と合った文字を検索しています。

stringオブジェクトの「match()」メソッドと似ていますが、この場合はドキュメントがアップデートされます。

```
<script language="JavaScript1.2">
re = new RegExp("Nav");
s4 = "NetscapeNavgator4.0";
exe = re.exec(s4);
document.write(exe);
</script>
```

付録コラム

# Signed Script（署名付きスクリプト）の使い方

Netscape Navigator 4.0で採用されているJavaScript1.2では、セキュリティなどの関係で、ディスプレイの表示領域外に新しいウィンドウを開くことができない、100×100ピクセル以下のサイズのウィンドウを新しく開くことができないなど、スクリプトの一部に制約をかけています。

こういった制約を超えてスクリプトを実行したい時や、スクリプトの動作前にユーザーに認証をしてもらいたい時などに、Java Archive（JAR）を利用して、ユーザーにそのスクリプトの開発元などの情報を提示し、ユーザーの承認を得てからスクリプトを実行させるということができます。この認証手続きの用法を「Signed Script」といいます。

Signed Scriptは、次の例のように<script>内で「archive」属性と「id」属性を指定して行い、その内の「archive」はデジタル認証のソースであるJava Archive（JAR）ファイル名を、「id」はJava Archive（JAR）ファイルからの署名と組み合わせて使う文字列を指定します。

```
<script archive="myArchive.jar" id="1">
document.write("これは、Signed Scriptです。");
</script>
```

イベントハンドラと組み合わせて使用する場合は、イベントハンドラが含まれるタグ内には「archive」を指定する必要はありませんが、次の例のようにイベントハンドラより前に「Signed Script」の指定を行っておく必要があります。

また、「Signed Script」では、「<a href="javascript:...」は使えません。

```
<script archive="myArchive.jar" id="1">
   ～
</script>
<form>
<input type="button" value="クリックして!!" onClick="alert('これはSigned
 Scriptです!!')" id="2">
</form>
```

「archive」属性で指定するJava Archive（JAR）ファイルは、1種類しか使用できません。始めのHTMLファイル中にJava Archive（JAR）ファイルがあった場合、それ以降の「Signed Script」で指定されたスクリプトは、次の例のように「id」を指定することにより、同じJava Archive（JAR）ファイルを使用するようにします。

この時、HTMLファイルが異なったプラットフォームのサーバー上にまたがって置かれているような場合（たとえばWindowsとUNIX間）、問題が発生する可能性があります。

```
<script archive="myArchive.jar" id="1">
document.write("これは、Signed Scriptです。");
</script>

<script id="2">
document.write("これも、Signed Scriptです。");
document.write("<br>");
document.write("myArchive.jarのデジタル認証を使用しています。");
</script>
```

「Signed Script」を含めたNetscape Navigator 4.0のセキュリティに関しては、

**「Client-Side JavaScript Guide」**
(http://tecfa.unige.ch/guides/js/jsguide13/index.html)

の「Chapter 14 JavaScript Security」などを参考にしてください(英文)。

# Live Connect

Netscape Navigator 3.0からは、JavaScriptを使ってブラウザ上のJAVAアプレットやプラグインを制御するための仕掛けが用意されました。この仕掛けを「Live Connect」と言います。

Live Connectを使用するためには、JAVAアプレットもプラグインもLive Connectに対応している必要があります。

Netscape社では、ブラウザ上からWebサーバーの設定を可能にするためにこのLive Connectの技術を使うなど、ブラウザを他の機能と結び付けるための接着剤の役割として使っており、今後ますます利用範囲が広がって行くと思われます。

Live Connectに関したJavaScript側で用意されているオブジェクトには、Appletオブジェクトとembeds配列があります。

## Appletオブジェクト（配列）

**【用法】**

```
document.applets[インデックス]
document.applets.length
```

**【プロパティ】**

| | |
|---|---|
| length | アプレットの数（配列数） |

**【メソッド】**

JAVAアプレットのすべての"public method"

| | |
|---|---|
| eval() | 文字列を数値に変える |
| toString() | オブジェクトを文字列に変える |
| valueOf() | オブジェクトの値を返す |

## embeds配列

**【用法】**

```
document.embeds[インデックス]
document.embeds.length
```

**【プロパティ】**

| | |
|---|---|
| length | プラグインの数（配列数） |

**【メソッド】**

各プラグイン独自のメソッド

# JavaScript作成支援ツール

この頃では、Webページ作成用のツールに、JavaScriptを使ったページを手軽に作成できる機能を搭載した製品が数多く発表されています。これらのほとんどが、設定画面から項目を選択したり必要な値を入れるだけで、Internet ExplorerとNetscapeのどちらでも実行できるJavaScriptのコードを自動的に生成することができます。

これらのツールは、自分でスクリプトを書くのと違い、できる範囲がそのツールの機能内に決まってしまいますが、手軽にJavaScriptを使ったページを作りたいようなユーザーには有効でしょう。また、これらのツールが生成するJavaScriptのコードを眺めるのも、JavaScriptのいい勉強になります。いくつか紹介しておきますので、参考にしてください。

### 「Adobe GoLive」

(http://www.adobe.co.jp/products/golive/main.html)

アドビシステムズ社のサイト管理機能も含む本格的なWebページ作成・管理ツール。JavaScriptをページ上に埋め込む機能の他、Internet Explorer・Netscape両ブラウザに対応したDynamicHTMLを使ったページを作成することが可能です。

### 「Adobe Photoshop」

(http://www.adobe.co.jp/products/photoshop/main.html)

アドビシステムズ社の画像編集ツール。Photoshop5.5から、Webページ用の画像編集ツールであるImageReadyが統合されており、両ソフトを切り替えて使用できるようになりました。ImageReadyでは、マウスカーソルが画像の上に乗った時に画像を変更するなどのJavaScriptを使った効果が用意されています。

### 「Macromedia Dreamweaver」

(http://www.macromedia.com/jp/software/dreamweaver/)

マクロメディア社のサイト管理機能も含む本格的なWebページ作成・管理ツール。JavaScriptをページ上に埋め込む機能の他、Internet Explorer・Netscape両ブラウザに対応したDynamicHTMLを使ったページを作成することが可能です。

### 「Macromedia Fireworks」

(http://www.macromedia.com/jp/software/fireworks/)

マクロメディア社のWebページ用の画像編集ツール。マウスカーソルが画像の上に乗った時に画像を変更するなどのJavaScriptを使った効果が用意されています。

### 「IBM ホームページ・ビルダー」

(http://www-6.ibm.com/jp/software/internet/hpb/index.html)

IBM社のサイト管理機能や画像・映像編集ツールも含む本格的なWebページ作成・管理ツール。JavaScriptをページ上に埋め込む機能の他、DynamicHTMLを使ったページの作成をしたり、画像作成時にマウスカーソルが画像の上に乗った時に画像を変更するなどのJavaScriptを使った効果を使うことが可能です。

# 索引



The revised edition of the detailed explanation dictionary
about HyperText Markup Language,
Cascading Style Sheets,
and JavaScript.

# 目的・機能別索引

## 数字



## アルファベット



## あ行

## か行



## さ行

## た行



## な行



## は行

目的・機能別索引

## ま行



## や行



## ら行



## わ行

# HTML索引

## 記号



## A



## B



## C



## D



## E～F

## H



## I



## K～L



## M～N

## O



## P～R



## S



## T



## U

# CSS索引

## 記号



## A



## B



## C



## D



## E



## F



## H



## L

## W



## Z



## 漢字

# JavaScript索引

## 記号



## A

## B



## C



## D



## E



## F



JavaScript索引

## G



## H



## I



## J



## K



## L

## M



## N



## O



JavaScript索引

## P



## R



## S

## T



## U



## V



## W

## X



## Y



## Z



## 漢字

# ホームページ作成用語索引

## 数字・記号



## A



## C



## D



## E



## F



## G



## H



## I

## J



## L



## M



## N



## O



## P



## R



## S



## T



## U



## W

## X



## あ行



## か行



ホームページ作成用語索引

## さ行

## た行

## な行



## は行

## ま行

## や行



## ら行



## わ行

# ホームページ作成関連リンク

## HTML関連リンク

### 「W3C - The World Wide Web Consortium」

URI http://www.w3.org/

World Wide Webの標準化作業を行っている非営利団体W3Cのサイト(英語)

### 「W3Cの仕様書等の文書の日本語訳集」

URI http://www.w3.org/Consortium/Translation/Japanese

W3C仕様書の日本語訳の一覧

### 「W3C HTML Validation Service」

URI http://validator.w3.org/

W3CによるHTMLの検証サービス(英語)

### 「HTML 4.01 Specification」

URI http://www.w3.org/TR/html4/

HTML4.01の仕様書の原文(英語)

### 「XHTML 1.0: The Extensible HyperText Markup Language」

URI http://www.w3.org/TR/xhtml1/

XHTML1.0の仕様書の原文(英語)

### 「Modularization of XHTML」

URI http://www.w3.org/TR/xhtml-modularization/

XHTMLのモジュール化の仕様書の原文(英語)

### 「XHTML Basic」

URI http://www.w3.org/TR/xhtml-basic/

XHTML Basicの仕様書の原文(英語)

### 「XHTML 1.1 - Module-based XHTML」

URI http://www.w3.org/TR/xhtml11/

XHTML1.1の仕様書の原文(英語)

### 「Ruby Annotation」

URI http://www.w3.org/TR/ruby/

ルビ(モジュール)の仕様書の原文(英語)

「The Web KANZAKI」

URI http://kanzaki.com/

HTMLを正しく理解するためのわかりやすい解説がある

【アクセシビリティ】

「Web Content Accessibility Guidelines 1.0」

URI http://www.w3.org/TR/WAI-WEBCONTENT/

W3Cの勧告であるアクセシビリティ・ガイドライン1.0の原文

「Techniques for Web Content Accessibility Guidelines 1.0」

URI http://www.w3.org/TR/WCAG10-TECHS/

上記ガイドラインについての、より具体的な解説とサンプルなど

「ZSPC」

URI http://www.zspc.com/

アクセシブルなウェブデザインに関する情報サイト

「A.A.O. - Webアクセシビリティ実用サイト」

URI http://www.aao.ne.jp/

アクセシブルなウェブをめざす提供者と利用者のための実用サイト

「Webアクセシビリティのポータルサイト『Infoaxia（インフォアクシア)』」

URI http://www.infoaxia.com/

Webアクセシビリティを理解し実践するためのポータルサイト

## CSS関連リンク

「Cascading Style Sheets, level 1」

URI http://www.w3.org/TR/REC-CSS1

CSS1の仕様書の原文(英語)

「Cascading Style Sheets, level 2」

URI http://www.w3.org/TR/REC-CSS2/

CSS2の仕様書の原文(英語)

「W3C CSS検証サービス」

URI http://jigsaw.w3.org/css-validator/

W3CによるCSSの検証サービス

「CSS2対応状況ガイド」

URI http://www.zspc.com/documents/css2/

各種ブラウザのCSS2の対応状況の一覧

## JavaScript関連リンク

### Netscape社のリファレンス

「JavaScript Guide」

URI http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html

JavaScript 1.1のガイド(英語)

### Microsoft社のリファレンス

「JScript」

URI http://msdn.microsoft.com/scripting/jscript/default.htm

Microsoft社のJavaScript互換のスクリプトJScriptの解説(英語)

「JScript」

URI http://www.microsoft.com/Japan/Developer/Scripting/JScript/default.htm

JScriptの日本語解説ページ

「HTML and Dynamic HTML」

URI http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/dhtml_node_entry.asp

Internet ExplorerにおけるDynamicHTMLの解説(英語)

### mozilla.orgの関連資料

「JavaScript」

URI http://www.mozilla.org/js/

mozilla.orgのJavaScript1.5関連情報のページ(英語)

「Gecko DOM reference」

URI http://www.mozilla.org/docs/dom/domref/

NetscapeやMozillaで実装されているレンダリングエンジン、GeckoのDOMリファレンス(英語)

### その他のJavaScript関連の仕様

「The World Wide Web Consortium」

URI http://www.w3.org/

Webの仕様の標準化を行っているW3Cのサイト。HTML、CSS、DOMなどの仕様を公開(英語)

「ECMA-262」

URI http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM

ECMAScript(ECMA-262)の仕様(英語)

### 「Core JavaScript Guide」

URI http://synchro.net/docs/js/guide/index.html
以前はNetscape社にあったJavaScript1.5のガイド(英語)

### 「Core JavaScript Reference 」

URI http://synchro.net/docs/js/ref/index.html
以前はNetscape社にあったJavaScript1.5のリファレンス(英語)

## メーリングリスト

### 「js-ML」

URI http://www.fureai.or.jp/~tato/js-ml/jsml.htm
DynamicHTMLおよびDHTMLcross-browser、JavaScript関連のメーリングリスト

## ネットニュース

### comp.lang.javascript

英語のJavaScriptの話題を取り扱ったニュースグループ(英語)

### fj.lang.javascript

日本語のJavaScriptの話題を取り扱ったニュースグループ

## その他のWebページ

### 「OpenSpace」

URI http://www.shiojiri.ne.jp/~openspc/
情報量は随一

### 「とほほのWWW入門」

URI http://tohoho.wakusei.ne.jp/
HTMLやJavaScriptをはじめとした、総合Webページ作成関連情報

### 「独学JavaScript」

URI http://www.ueda.info.waseda.ac.jp/~gaku/js/index.html
JavaScript関連ページの老舗

### 「JavaScript Tips集」

URI http://www.din.or.jp/~hagi3/JavaScript/JSTips/Default.htm
JavaScript、DHTML関連、cross-browserDHTML情報が多数あり

### 「WebPageで使えるJavaScript」

URI http://www.bekkoame.ne.jp/~hamba/webimage/java/java.html
JavaScript関連リンクが充実している

### 「All About Japan [JavaScript]」

URI http://allabout.co.jp/computer/javascript/
JavaScript関連リンクをはじめ、多くの関連情報がある

### 「Mozilla's DOM Sample Project」

URI http://cgi.din.or.jp/~hagi3/JavaScript/JSTips/Mozilla/mds.cgi
Mozillaで使用できるDOMのサンプルを集めたサイト(英語)

**コラム**

### 「JavaScript Developer Central」はどこへ?

Netscape社のサイトには、開発者向けのページとして、「DevEdge Online」(http://developer.netscape.com/)というページが用意されて、一般に公開されていました。そこには、JavaScriptに関する各種仕様書やサンプル、ガイドを収めた「JavaScript Developer Central」(http://developer.netscape.com/tech/javascript/index.html)のページも含まれていました。

しかし、2004年12月初旬から、この開発者向けのページにアクセスできなくなってしまいました。そして2005年2月初旬現在でも、アクセスできない状態が続いています。

確かに、Netscape Navigatorは、Netscape 8の開発者向けリリースが行われたこともあり、開発としては区切りが来ていたと言えます。また、一般に公開された形の開発者向けのページが、このままなくなってしまうとは思えません。

しかし、「JavaScript Developer Central」は、JavaScriptが、バージョンごとにどのように変化·発展してきたかを知ることができ、JavaScriptの歴史がすべて残されていると言ってもよい貴重なページでした。このページが、このままなくなってしまうとしたら、残念なことです。

# 参考書籍

JavaScriptやDynamicHTMLを知るために役立ちそうな書籍や、著者が普段から参考にしている書籍を紹介します。

## HTML&CSS関連書籍

**「XHTML+CSSで書くホームページ構造デザインガイド」**
（大藤幹:著　秀和システム刊）

**「スタイルシートサンプルブック」**（大藤幹:著　ソシム刊）

**「詳解HTML&XHTML&CSS辞典 改訂版」**（大藤幹:著　秀和システム刊）

## JavaScript関連書籍

**「フルカラー版だれでもカンタンJavaScriptサンプル集」**
（高橋登史朗:著　秀和システム刊）

**「ちょ〜初心者のためのJavaScript入門」**（半場方人:著　秀和システム刊）

**「JavaScript入門」**（半場方人:著　技術評論社刊）

**「改定第2版JavaScriptポケットリファレンス」**（古籏一浩:著　技術評論社刊）

**「OFFICIAL Netscape JavaScript」**（Peter Kent, John Kent:著　BNN刊）

## DynamicHTML関連書籍

**「詳解JavaScript&DynamicHTML辞典」**（半場方人:著　秀和システム刊）

**「ダイナミックHTMLファーストステップ」**（松尾忠則:著　秀和システム刊）

**「だれでもカンタンDynamicHTMLサンプル集」**（高橋登史朗:著　秀和システム刊）

**「DynamicHTMLでつくるホームページ」**
（松尾忠則・半場方人・すぎうらしろう:著　古籏一浩:監修　技術評論社刊）

**「使えるJavaScript&DynamicHTMLサンプル大全」**（萩原真一:著　秀和システム刊）

**「ここまでできる!JavaScriptらくらく活用サンプル集」**（高橋登史朗:著　秀和システム刊）

**本書のサポートページ**

http://www.shuwasystem.co.jp/SHOKAIdic/

詳解（しょうかい）
HTML&CSS&JavaScript辞典（エイチティーエムエルアンドシーエスエスアンドジャバスクリプトじてん）
改訂版（かいていばん）

発行日　2005年　4月10日　　第1版第1刷

著　者　大藤（おおふじ）　幹（みき）／半場（はんば）　方人（まさひと）



発行者　牧谷　秀昭
発行所　株式会社　秀和システム
〒107-0062　東京都港区南青山1-26-1 寿光ビル5F
Tel 03-3470-4947(販売)
Fax 03-3405-7538
印刷所　三松堂印刷株式会社


ISBN4-7980-1005-7 C3055

## カラーチャート1：HTML4.01で名前が定義されている色

HTML4.01でcolor属性の値として定義されている色名は、以下の通りです(色の名前には大文字と小文字の区別はありません)。

これらは、CSS2でも同様にプロパティの値として利用できます。現在では、HTMLの属性で色を指定することは非推奨とされていますので、できるだけスタイルシートを使用するようにしてください。

具体的な色の指定の方法は、HTMLの場合は「HTMLについて」の「色の指定方法」(P.6)、CSSの場合は「スタイルシートについて」の「色の指定方法」(P.180)を参照してください。

| | | | |
|---|---|---|---|
| #000000 | black | #008000 | green |
| #c0c0c0 | silver | #00ff00 | lime |
| #808080 | gray | #808000 | olive |
| #ffffff | white | #ffff00 | yellow |
| #800000 | maroon | #000080 | navy |
| #ff0000 | red | #0000ff | blue |
| #800080 | purple | #008080 | teal |
| #ff00ff | fuchsia | #00ffff | aqua |

## Web Safeカラーについて

パソコンの画面上の色は、RGB(Red、Green、Blue)の色をそれぞれ0～255までの256段階に調節して色を表現しています。したがって、フルカラーでは256×256×256(R×G×B)の16,777,216色が表現されます。

「Web Safeカラー」とは、この各256段階あるRGB値を6段階にしたものです。そして、その6段階の値とは、0・51・102・153・204・255と、0から51ずつ増やしていった数になっています。51は16進数でいうと「33」、割合でいうと「20％」にあたる数です。つまり、「Web Safeカラー」は、RGBの各値(画面上では光の強さ)を0%から20%ずつ上げていった値の組み合せで作られているということになります。

| | | | | | | |
|---|---|---|---|---|---|---|
| 10進数 | 0 | 51 | 102 | 153 | 204 | 255 |
| 16進数 | 0 | 33 | 66 | 99 | cc | ff |
| % | 0% | 20% | 40% | 60% | 80% | 100% |

これらを組み合わせると、RGBの3色がそれぞれ6段階あるので、6×6×6の216色がWeb Safeカラーということになります。

具体的な色については、次ページを参照してください。

# カラーチャート2：Web Safeカラー

WindowsやMacintoshなどのプラットフォームでそれぞれ設定されている、256色のシステムカラーパレットのうち、共通している216色のカラーチャートです。

これらの色を使うことで、異なるOS環境でも色化けなどを生じさせることなく表示することができます。

印刷された色は、実際に画面に表示されている色とは多少異なります。大体の色の感じを掴むためにご利用ください。

| | | | | | |
|---|---|---|---|---|---|
| #000000 (000,000,000) | #003300 (000,051,000) | #006600 (000,102,000) | #009900 (000,153,000) | #00cc00 (000,204,000) | #00ff00 (000,255,000) |
| #000033 (000,000,051) | #003333 (000,051,051) | #006633 (000,102,051) | #009933 (000,153,051) | #00cc33 (000,204,051) | #00ff33 (000,255,051) |
| #000066 (000,000,102) | #003366 (000,051,102) | #006666 (000,102,102) | #009966 (000,153,102) | #00cc66 (000,204,102) | #00ff66 (000,255,102) |
| #000099 (000,000,153) | #003399 (000,051,153) | #006699 (000,102,153) | #009999 (000,153,153) | #00cc99 (000,204,153) | #00ff99 (000,255,153) |
| #0000cc (000,000,204) | #0033cc (000,051,204) | #0066cc (000,102,204) | #0099cc (000,153,204) | #00cccc (000,204,204) | #00ffcc (000,255,204) |
| #0000ff (000,000,255) | #0033ff (000,051,255) | #0066ff (000,102,255) | #0099ff (000,153,255) | #00ccff (000,204,255) | #00ffff (000,255,255) |
| #330000 (051,000,000) | #333300 (051,051,000) | #336600 (051,102,000) | #339900 (051,204,000) | #33cc00 (051,204,000) | #33ff00 (051,255,000) |
| #330033 (051,000,051) | #333333 (051,051,051) | #336633 (051,102,051) | #339933 (051,153,051) | #33cc33 (051,204,051) | #33ff33 (051,255,051) |
| #330066 (051,000,102) | #333366 (051,051,102) | #336666 (051,102,102) | #339966 (051,153,102) | #33cc66 (051,204,102) | #33ff66 (051,255,102) |
| #330099 (051,000,153) | #333399 (051,051,153) | #336699 (051,102,153) | #339999 (051,153,153) | #33cc99 (051,204,153) | #33ff99 (051,255,153) |
| #3300cc (051,000,204) | #3333cc (051,051,204) | #3366cc (051,102,204) | #3399cc (051,153,204) | #33cccc (051,204,204) | #33ffcc (051,255,204) |
| #3300ff (051,000,255) | #3333ff (051,051,255) | #3366ff (051,102,255) | #3399ff (051,153,255) | #33ccff (051,204,255) | #33ffff (051,255,255) |
| #660000 (102,000,000) | #663300 (102,051,000) | #666600 (102,102,000) | #669900 (102,153,000) | #66cc00 (102,204,000) | #66ff00 (102,255,000) |
| #660033 (102,000,051) | #663333 (102,051,051) | #666633 (102,102,051) | #669933 (102,153,051) | #66cc33 (102,204,051) | #66ff33 (102,255,051) |
| #660066 (102,000,102) | #663366 (102,051,102) | #666666 (102,102,102) | #669966 (102,153,102) | #66cc66 (102,204,102) | #66ff66 (102,255,102) |

| | | | | | |
|---|---|---|---|---|---|
| #660099 (102,000,153) | #663399 (102,051,153) | #666699 (102,102,153) | #669999 (102,153,153) | #66cc99 (102,204,153) | #66ff99 (102,255,153) |
| #6600cc (102,000,204) | #6633cc (102,051,204) | #6666cc (102,102,204) | #6699cc (102,153,204) | #66cccc (102,204,204) | #66ffcc (102,255,204) |
| #6600ff (102,000,255) | #6633ff (102,051,255) | #6666ff (102,102,255) | #6699ff (102,153,255) | #66ccff (102,204,255) | #66ffff (102,255,255) |
| #990000 (153,000,000) | #993300 (153,051,000) | #996600 (153,102,000) | #999900 (153,153,000) | #99cc00 (153,204,000) | #99ff00 (153,255,000) |
| #990033 (153,000,051) | #993333 (153,051,051) | #996633 (153,102,051) | #999933 (153,153,051) | #99cc33 (153,204,051) | #99ff33 (153,255,051) |
| #990066 (153,000,102) | #993366 (153,051,102) | #996666 (153,102,102) | #999966 (153,153,102) | #99cc66 (153,204,102) | #99ff66 (153,255,102) |
| #990099 (153,000,153) | #993399 (153,051,153) | #996699 (153,102,153) | #999999 (153,153,153) | #99cc99 (153,204,153) | #99ff99 (153,255,153) |
| #9900cc (153,000,204) | #9933cc (153,051,204) | #9966cc (153,102,204) | #9999cc (153,153,204) | #99cccc (153,204,204) | #99ffcc (153,255,204) |
| #9900ff (153,000,255) | #9933ff (153,051,255) | #9966ff (153,102,255) | #9999ff (153,153,255) | #99ccff (153,204,255) | #99ffff (153,255,255) |
| #cc0000 (204,000,000) | #cc3300 (204,051,000) | #cc6600 (204,102,000) | #cc9900 (204,153,000) | #cccc00 (204,204,000) | #ccff00 (204,255,000) |
| #cc0033 (204,000,051) | #cc3333 (204,051,051) | #cc6633 (204,102,051) | #cc9933 (204,153,051) | #cccc33 (204,204,051) | #ccff33 (204,255,051) |
| #cc0066 (204,000,102) | #cc3366 (204,051,102) | #cc6666 (204,102,102) | #cc9966 (204,153,102) | #cccc66 (204,204,102) | #ccff66 (204,255,102) |
| #cc0099 (204,000,153) | #cc3399 (204,051,153) | #cc6699 (204,102,153) | #cc9999 (204,153,153) | #cccc99 (204,204,153) | #ccff99 (204,255,153) |
| #cc00cc (204,000,204) | #cc33cc (204,051,204) | #cc66cc (204,102,204) | #cc99cc (204,153,204) | #cccccc (204,204,204) | #ccffcc (204,255,204) |
| #cc00ff (204,000,255) | #cc33ff (204,051,255) | #cc66ff (204,102,255) | #cc99ff (204,153,255) | #ccccff (204,204,255) | #ccffff (204,255,255) |
| #ff0000 (255,000,000) | #ff3300 (255,051,000) | #ff6600 (255,102,000) | #ff9900 (255,153,000) | #ffcc00 (255,204,000) | #ffff00 (255,255,000) |
| #ff0033 (255,000,051) | #ff3333 (255,051,051) | #ff6633 (255,102,051) | #ff9933 (255,153,051) | #ffcc33 (255,204,051) | #ffff33 (255,255,051) |
| #ff0066 (255,000,102) | #ff3366 (255,051,102) | #ff6666 (255,102,102) | #ff9966 (255,153,102) | #ffcc66 (255,204,102) | #ffff66 (255,255,102) |
| #ff0099 (255,000,153) | #ff3399 (255,051,153) | #ff6699 (255,102,153) | #ff9999 (255,153,153) | #ffcc99 (255,204,153) | #ffff99 (255,255,153) |
| #ff00cc (255,000,204) | #ff33cc (255,051,204) | #ff66cc (255,102,204) | #ff99cc (255,153,204) | #ffcccc (255,204,204) | #ffffcc (255,255,204) |
| #ff00ff (255,000,255) | #ff33ff (255,051,255) | #ff66ff (255,102,255) | #ff99ff (255,153,255) | #ffccff (255,204,255) | #ffffff (255,255,255) |

# カラーチャート3：Color Name

HTML4.01・CSS2で正式に設定されている色、プラットフォーム間で共通な色、というように紹介してきましたが、最後はブラウザで設定されている色です。

以下のチャートは、Internet ExplorerとNetscape Navigatorで名前が定義されている140色です（HTML4.01で定義されている16色も含まれています）。色を名前で設定する時に利用することができますが、あくまでもブラウザが対応しているものであるので、不安定な部分があることに注意してください。

印刷された色は、実際に画面に表示されている色とは多少異なります。大体の色の感じを掴むためにご利用ください。

| | |
|---|---|
| #ffffff | White |
| #fffafa | Snow |
| #f8f8ff | GhostWhite |
| #f5f5f5 | WhiteSmoke |
| #fffaf0 | FloralWhite |
| #faf0e6 | Linen |
| #faebd7 | AntiqueWhite |
| #ffefd5 | PapayaWhip |
| #ffebcd | BlanchedAlmond |
| #ffe4c4 | Bisque |
| #ffe4b5 | Moccasin |
| #ffdead | NavajoWhite |
| #ffdab9 | PeachPuff |
| #ffe4e1 | MistyRose |
| #fff0f5 | LavenderBlush |
| #fff5ee | Seashell |
| #fdf5e6 | OldLace |
| #fffff0 | Ivory |
| #f0fff0 | Honeydew |
| #f5fffa | MintCream |
| #f0ffff | Azure |
| #f0f8ff | AliceBlue |
| #e6e6fa | Lavender |
| #000000 | Black |
| #2f4f4f | DarkSlateGray |
| #696969 | DimGray |
| #808080 | Gray |
| #a9a9a9 | Darkgray |
| #c0c0c0 | Silver |
| #d3d3d3 | LightGrey |
| #dcdcdc | Gainsboro |
| #778899 | LightSlateGray |
| #708090 | SlateGray |
| #b0c4de | LightSteelBlue |
| #4682b4 | SteelBlue |
| #4169e1 | RoyalBlue |
| #191970 | MidnightBlue |
| #000080 | Navy |
| #00008b | Darkblue |
| #0000cd | MediumBlue |
| #0000ff | Blue |
| #1e90ff | DodgerBlue |
| #6495ed | CornflowerBlue |
| #00bfff | DeepSkyBlue |
| #87cefa | LightSkyBlue |
| #87ceeb | SkyBlue |
| #add8e6 | LightBlue |
| #b0e0e6 | PowderBlue |
| #afeeee | PaleTurquoise |
| #e0ffff | LightCyan |
| #00ffff | Cyan |
| #00ffff | Aqua |
| #40e0d0 | Turquoise |
| #48d1cc | MediumTurquoise |
| #00ced1 | DarkTurquoise |
| #20b2aa | LightSeaGreen |

| | |
|---|---|
| #5f9ea0 | CadetBlue |
| #008b8b | Darkcyan |
| #008080 | Teal |
| #2e8b57 | SeaGreen |
| #556b2f | DarkOliveGreen |
| #006400 | DarkGreen |
| #008000 | Green |
| #228b22 | ForestGreen |
| #3cb371 | MediumSeaGreen |
| #8fbc8f | DarkSeaGreen |
| #66cdaa | MediumAquamarine |
| #7fffd4 | Aquamarine |
| #98fb98 | PaleGreen |
| #90ee90 | LightGreen |
| #00ff7f | SpringGreen |
| #00fa9a | MediumSpringGreen |
| #7cfc00 | LawnGreen |
| #7fff00 | Chartreuse |
| #adff2f | GreenYellow |
| #00ff00 | Lime |
| #32cd32 | LimeGreen |
| #9acd32 | YellowGreen |
| #6b8e23 | OliveDrab |
| #808000 | Olive |
| #bdb76b | DarkKhaki |
| #eee8aa | PaleGoldenrod |
| #fff8dc | Cornsilk |
| #f5f5dc | Beige |
| #ffffe0 | LightYellow |
| #fafad2 | Lightgoldenrodyellow |
| #fffacd | LemonChiffon |
| #f5deb3 | Wheat |
| #deb887 | Burlywood |
| #d2b48c | Tan |
| #f0e68c | Khaki |
| #ffff00 | Yellow |
| #ffd700 | Gold |
| #ffa500 | Orange |
| #f4a460 | SandyBrown |
| #ff8c00 | DarkOrange |
| #daa520 | Goldenrod |
| #cd853f | Peru |

| | |
|---|---|
| #b8860b | DarkGoldenrod |
| #d2691e | Chocolate |
| #a0522d | Sienna |
| #8b4513 | SaddleBrown |
| #800000 | Maroon |
| #8b0000 | DarkRed |
| #a52a2a | Brown |
| #b22222 | Firebrick |
| #cd5c5c | IndianRed |
| #bc8f8f | RosyBrown |
| #e9967a | DarkSalmon |
| #f08080 | LightCoral |
| #fa8072 | Salmon |
| #ffa07a | LightSalmon |
| #ff7f50 | Coral |
| #ff6347 | Tomato |
| #ff4500 | OrangeRed |
| #ff0000 | Red |
| #dc143c | Crimson |
| #c71585 | MediumVioletRed |
| #ff1493 | DeepPink |
| #ff69b4 | HotPink |
| #db7093 | PaleVioletRed |
| #ffc0cb | Pink |
| #ffb6c1 | LightPink |
| #d8bfd8 | Thistle |
| #ff00ff | Magenta |
| #ff00ff | Fuchsia |
| #ee82ee | Violet |
| #dda0dd | Plum |
| #da70d6 | Orchid |
| #ba55d3 | MediumOrchid |
| #9932cc | DarkOrchid |
| #9400d3 | DarkViolet |
| #8b008b | DarkMagenta |
| #800080 | Purple |
| #4b0082 | Indigo |
| #483d8b | DarkSlateBlue |
| #8a2be2 | BlueViolet |
| #9370db | MediumPurple |
| #6a5acd | SlateBlue |
| #7b68ee | MediumSlateBlue |

# フォント表示見本

## Windows

### MSゴシック
日本語漢字見本
あいうえおアイウエオ
0123456789 ! ″ # $ % & / * - +

### MS Pゴシック
日本語漢字見本
あいうえおアイウエオ
0123456789 ! ″ # $ % & / * - +

### MS 明朝
日本語漢字見本
あいうえおアイウエオ
0123456789 ! ″ # $ % & / * - +

### MS P明朝
日本語漢字見本
あいうえおアイウエオ
0123456789 ! ″ # $ % & / * - +

### Arial
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz<
0123456789 ! " # $ % & / * - +

### Century Gothic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

### Comic Sans MS
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

### Copperplate
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789 ! " # $ % & / * - +

### Courier
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

### Courier New
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

### Impact
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

### Lucida
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

### News Gothic
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

### Symbol
ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟΠΘΡΣΤΥςΩΞΨΖ
αβχδεφγηιϕκλμνοπθρστυϖωξψζ
0123456789 ! ∀ # ∃ % & / ∗ − +

### Times New Roman
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

### Verdana
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

WindowsとMacintoshで最初からOSにインストールされているフォントをブラウザで表示させました。フォント名を指定する時の参考にしてください。

## Macintosh

**Osaka**

日本語漢字見本
あいうえおアイウエオ
0123456789 ! " # $ % & / * - +

**Osaka-等幅**

日本語漢字見本
あいうえおアイウエオ
0123456789 ! " # $ % & / * - +

**細明朝体**

日本語漢字見本
あいうえおアイウエオ
0123456789 ! " # $ % & / * - +

**中ゴシック体**

日本語漢字見本
あいうえおアイウエオ
0123456789 ! " # $ % & / * - +

**平成角ゴシック**

日本語漢字見本
あいうえおアイウエオ
0123456789 ! " # $ % & / * - +

**平成明朝**

日本語漢字見本
あいうえおアイウエオ
0123456789 ! " # $ % & / * - +

**Charcoal**

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

**Chicago**

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

**Courier**

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

**Geneva**

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

**Helvetica**

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

**Monaco**

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

**New York**

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

**Palatino**

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

**Symbol**

ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟΠΘΡΣΤΥςΩΞΨΖ
αβχδεφγηιφκλμνοπθρστυϖωξψζ
0123456789 ! ∀ # ∃ % & / * - +

**Times**

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 ! " # $ % & / * - +

# Webサイズチャート

HTML・CSS・JavaScriptでサイズを指定する時に使う単位を解説します。ただし、px・%以外は、スタイルシート内でしか使うことができません。

## 相対的な大きさの単位

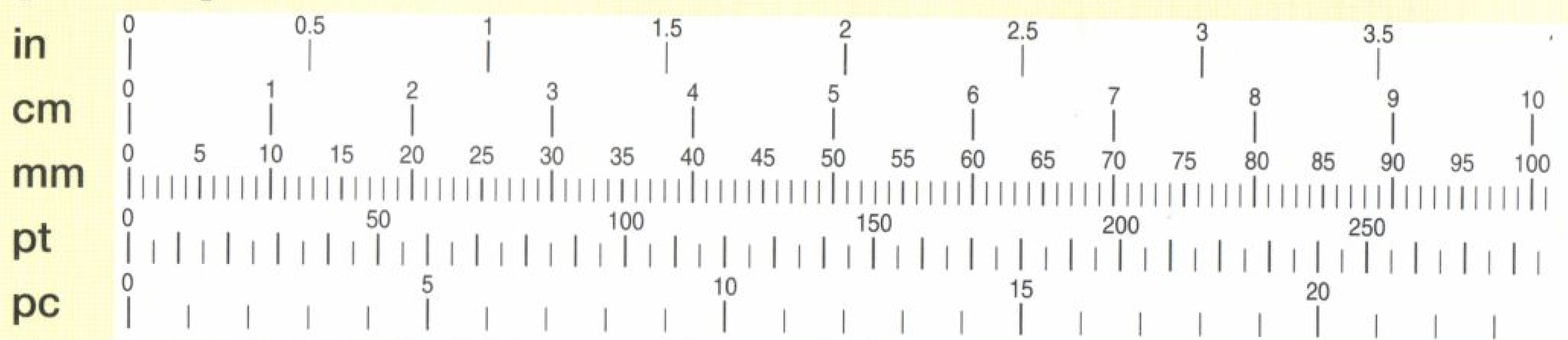
- **px**　1ピクセルを1とする単位です。主にコンピュータのディスプレイを出力対象としますが、その解像度によって表示される大きさが異なってきますので、注意してください。
- **%**　他の大きさに対する割合で表す場合に使用します。対象とする大きさは状況に応じて、同じ要素内の他のプロパティ、親要素のプロパティ、それを含むブロックの幅などになります。
- **em**　その範囲で有効なフォントの高さ(font-sizeの値)を1とする単位です。スタイルシートのfont-sizeプロパティの値としてこの単位が使用された場合には、親要素で有効なフォントの高さを1とする値になります。
- **ex**　その範囲で有効なフォントのアルファベットの小文字「x」の高さを1とする単位です。フォントによっては、必ずしも「x」の高さと一致するとは限りません。また、この単位は「x」を含まないフォントにも適用されます。




## 絶対的な大きさの単位

- **in**　インチ。1インチは2.54cm(=25.4mm)です。
- **cm**　センチメートル。1センチメートルは10ミリメートルです。
- **mm**　ミリメートル。1ミリメートルは1/10センチメートルです。
- **pt**　ポイント。1ポイントは1/72インチです。一般的に文字のサイズを指定する時に使います。
- **pc**　パイカ。1パイカは12ポイントです。

**【メジャー】**




**【文字サイズ】**

**8pt**　あいうえおABCabc1234567890

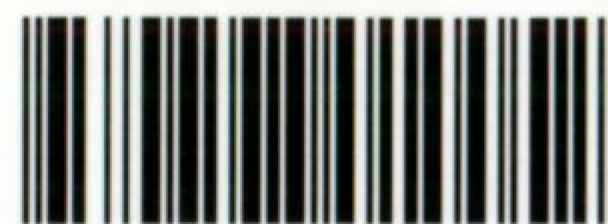
**12pt**　あいうえおABCabc1234567890

**18pt**　あいうえおABCabc1234567890

**24pt**　あいうえおABCabc123456789

ISBN4-7980-1005-7

C3055 ¥2100E







定価　(本体 2100円+税)

The revised edition of the detailed explanation dictionary
about HyperText Markup Language,
Cascading Style Sheets,
and JavaScript.

秀和システム

詳解 HTML & CSS & JavaScript 辞典 改訂版

大藤 幹
半場 方人 著

